Università degli Studi di Trieste

XXX Ciclo del Dottorato di Ricerca in

Ingegneria dell'Informazione

VEHICLE ROUTING PROBLEMS: DECISION SUPPORT SYSTEMS AND DISTRIBUTED APPROACHES

Settore scientifico-disciplinare: MAT/09

Dottorando:
Lorenzo ABBATECOLA

Coordinatore:
Chiar.mo Prof. Diego MICHELI
(Università degli Studi di Trieste)

Supervisore di Tesi:
Chiar.mo Prof. Walter UKOVICH
(Università degli Studi di Trieste)

Co-supervisore di Tesi:
Chiar.ma Prof.ssa Maria Pia FANTI
(Politecnico di Bari)

Anno Accademico 2016-2017

Contents

Chapter 1

Introduction

Modern logistics received increasing attention for planning and scheduling operations of transport systems that have to be resource efficient, environmentally sustainable, and compatible with workers' rights. In particular, modern timeliness requirements and technological advances respectively call for and enable new formulations and solutions for the classical Vehicle Routing Problem (VRP).

The classical and *static* VRP consists in assigning routes, at minimal cost, to a fleet of vehicles in order to serve a set of customers whose locations and demands are usually known in advance and do not change afterward. Typically, the solution is an *a priori* routing plan in which each vehicle departs from a central depot, serves a subset of customers compatibly with its capacity, and finally returns to the depot. In order to model real-world applications and their related different details (pick-up and delivery, multi-depots, multi-travels, work shifts, time windows, etc.), a great number of VRP variants have been introduced and intensively studied in literature. A scheme about taxonomy is available in [1], with respect to scenario characteristics and physical problem characteristics, and short descriptions about variants can be found in [2].

Most of these variants of VRP can be classified as *static* and *deterministic* [3] because they deal with an environment where all information is well known at the initial planning stage and remains static during the plan execution. Actually, in the real world, information may be affected by uncertainty and may be not completely available *a pri-*

ori because many unexpected events are likely to occur during plan execution, such as ongoing service requests, detection of actual amounts of demands just when vehicles are at customers' places, cancellation of orders, breakdown of vehicles and traffic congestion. Therefore, although having an initial plan could be generally useful, it needs of subsequent adaptations, while vehicles are performing it. In literature, a VRP facing an environment in which information is dynamically revealed to the decision maker is classified as *Dynamic* VRP (DVRP) [4] [3] [5]. **Chapter 2** is devoted to the description of DVRP and to a survey on recent literature.

The first review paper about DVRP dates back 1985 [4], but a renewed interest in this specific topic has steadily grown during the last decade due to recent economic and technological developments [5]. On one hand, logistics companies compete in a market in which, for example, customers make requests at every hour and call for an *as soon as possible* service. Thus, the availability of real-time information is crucial to be competitive. On the other hand, nowadays this real-time information (e.g., vehicle position and status, third party information about traffic, etc.) are made available, at low costs, by modern Information Communication Technologies (ICT), such as Intelligent Transport Systems (ITS) and highly spread on-board Global Positioning System (GPS) devices, as well as smart-phones. A list of these technologies is presented in [6], in which other advances are cited, such as speedups in computing hardware and commercial Mixed Integer Programming (MIP) solvers. Given these advances in technologies, a challenging issue for researchers is the elaboration of solving-problem methods able to interact with dynamic environments, i.e., proposing approaches that integrate real-time information and combine fast responses with solution quality. Significant contributions are reviewed and classified in two recent survey papers [3] and [6].

An important aspect related to advances in technologies, as well as to the complexity and the urgency character of real-world application, is the possibility and the requirement of inserting a vehicle routing planner module into a Decision Support Systems (DSS). Hence, **Chapter 3** presents a Urban DSS devoted to critical services in city logistics such as waste collection. The core of the system is a Vehicle Assignment and Routing module that can be fed with data tailored on different scenarios. The module is based

on a fast centralized heuristic algorithm that can be customized for different logistics services. The algorithm includes a two-phase heuristic able to solve a big instances of VRP with work shift constraints and has been assessed on the waste collection service for a city in the Northern Italy.

Returning to the state of the art, a thing that emerges reading survey papers about VRP and DVRP is that almost the whole presented research considers centralized approaches, in which a central dispatcher makes decision and communicates them to vehicles. There is little research on distributed and decentralized approaches, where vehicles are able to compute their respective routes by communicating with each other. In addition, most of the time, the optimization regards aspects that are convenient for the fleet owner and/or the customers, ignoring fairness of drivers' workloads.

Therefore, **Chapter 4** proposes a distributed approach for VRP with time windows constraints in a static and dynamic setting, which also takes in account workload balancing. The proposed approach relies on a *cluster first, route second* strategy where the clustering phase, i.e., assigning customers to vehicles, is modeled as a graph partitioning problem, extended with constraints on time length of travels and customers' time windows. By exploiting intrinsic balancing properties of the graph partitioning, the obtained clusters allow the subsequent routing phase to provide balanced routes with respect to traveling times and assigned loads. The routing phase, indeed, consists in computing the sequence of visits that a vehicle has to perform into its assigned cluster of customers.

About the distributed fashion of the proposed approach, separation of phases of the *cluster first, route second* method allows the routing phase to be already performed by each vehicle autonomously, i.e., each vehicle can solve an instance of Traveling Salesman Problem with Time Windows (TSPTW) autonomously, on its cluster. For the clustering phase, instead, a strategy similar to those presented in [7] and [8] is followed. More precisely, the clustering phase is performed by a distributed iterative algorithm that involves local instances of graph partitioning problem, in the form of local Integer Linear Programming (ILP) problems, and that enables vehicles, which act as intelligent agents, to autonomously and iteratively find a feasible (if it exists) solution of the assignment problem. In fact, starting from an initial arbitrary assignment, at each iteration, a vehicle

is chosen at random and it solves a local ILP problem involving only neighbor vehicles and their momentary assigned customers, which represent their current assignment state. As a consequence, the states of all the involved vehicles are updated. At the end of a certain number of non-changing iterations, the final solution of the global assignment problem is achieved and each vehicle can solve a TSPTW instance autonomously.
The distributed approach is applied to a VRP with Time Windows (VRPTW) and to a multi-depot VRPTW (MDVRPTW) that act as components of a DVRP, belonging to the *dynamic-deterministic* VRP class [3]: the former is the problem to be solved in order to obtain the initial routing plan; the latter can be seen as the problem the vehicles have to solve, at a certain point of the plan execution, in order to adapt the initial plan to the arrival of new customer requests, when the vehicles are in locations that can be thought as virtual departing depots for the re-optimization.
Comparisons with benchmark problems and the application to an example, inspired by a transport company, assess the effectiveness of the proposed distributed approach.

Chapter 2

Dynamic Vehicle Routing Problem: review of recent advances

This chapter reviews the state-of-the-art of one of the main problems about logistic issues, i.e., the Dynamic Vehicle Routing Problem. In particular, DVRP consists in assigning routes at minimal costs to a fleet of vehicles in order to serve a set of customers by also considering the information evolution (dynamism) and uncertainty (stochastic knowledge).

2.1 Introduction

The classical and *static* VRP deals with a *deterministic* operational environment where all information is well known before optimization (at planning time) and remains static during the execution of the routing plan [2], so no re-optimization is needed during the real execution of the plan. In the related literature this kind of VRP problem is referred as *static-deterministic* VRP [3].

Actually, in the real-world, information may be affected by uncertainty and may be not always completely available because of unexpected or uncertain events, such as on-going customer requests, actual amount of demand, breakdown of vehicles and traffic congestion. Thus, following [4] and [3], real-world applications have to consider two important dimensions of the information for the VRP: information *evolution* (static or

dynamic) and information *quality* (deterministic or stochastic). Combinations of those dimensions identify four types of VRPs: static-deterministic, static-stochastic, dynamic-deterministic and dynamic-stochastic.

This chapter focuses on the DVRP that includes dynamic-deterministic and dynamic-stochastic information. In *dynamic-deterministic* problems, part or all of the input is unknown and revealed dynamically during the design or the execution of the routes. In other word, new events occurs over time, so no definitive solution is constructed *a priori*, but routes are built in an ongoing fashion. Real-time communication between dispatcher (decision maker) and drivers/vehicles, or among decision agents, is supposed to exist.

The dynamically revealed input is called *source of dynamism*, and the most studied is the online arrival of customer requests. In *dynamic-stochastic* problems, stochastic knowledge is available on the source of dynamism and it is exploited to improve solutions. For instance, assuming the arrival of customer requests as source of dynamism, *proactive* strategies can lead vehicles towards request-likely areas. Otherwise, only *reactive* strategies can be applied [9] [10].

Together with real-world requirements, recent developments in ICT have accelerated the research on the DVRP [11] [6] by stressing challenges and promising developments in automation and decision making areas.

In order to enlighten the promising perspectives in the logistics management problems, this chapter presents a review about the recent contributions in the DVRP solutions. Starting from the survey paper by Pillac *et al.* of 2013 [3], we discuss some of the recent advances in DVRP and new paradigms such as distributed approaches, which are usually omitted by survey papers such as [6] and [3] itself.

The chapter is organized as follows. Section 2.2 introduces the DVRP with its peculiarities and challenges. Section 2.3 shows some DVRP applications, and Section 2.4 classifies the reviewed papers on the basis of solution approaches.

2.2 Dynamic Vehicle Routing Problem

2.2.1 The DRVP Definition

About origin of DRVP, Pillac *et al.* [3] date back it to Wilson and Colvin, in 1977 [12], and to Psaraftis [13], 1980, who introduced the concept of *immediate request*: a customer requiring immediate replanning of the current vehicle route.

According to Psaraftis [4], a VRP is *dynamic* if information (input data) on the problem is made known to the decision maker, or is updated, *concurrently* with the determination of the routes. For example, the typically studied DVRP concerns the online arrival of customer requests.

As a consequence, a continuous reoptimization process is needed and two concurrent levels have to be considered in DVRP [14] [15]: i) a planning level, which involves optimization and reoptimization; ii) an execution level, which involves the realization of the (under construction) plan in the dynamic environment (vehicle positions, new requests, other unexpected events).

In order to underline the ongoing fashion (the concurrency between optimization and execution) and the need of real-time communications, an example of centralized approach is showed in Fig. 2.1 and Fig. 2.2, inspired by [3] and [14]. Considering a single vehicle, Fig. 2.1 shows the results of progressive adjustments to the route as new requests arrive. Meanwhile, Fig. 2.2 shows communications between the vehicle and the dispatcher: the first gives its position and status; the second, which also collects requests, works out the best next destination and notifies the vehicle of it. In instant t_0 the vehicle is ready at the depot and the dispatcher decides to send it to customer A, on the basis of the best initial plan involving the known customers A, B and C. While the vehicle is traveling towards A, in instant t_1 the dispatcher receives a new request R and updates a theoretical plan accordingly. When the vehicle notifies the dispatcher about end of service at customer A, the dispatcher elaborates updated information and gives the vehicle its next destination R.

The most studied source of dynamism is the online arrival of new requests during the operations. Other sources of dynamism, related to customers, are the variation of de-

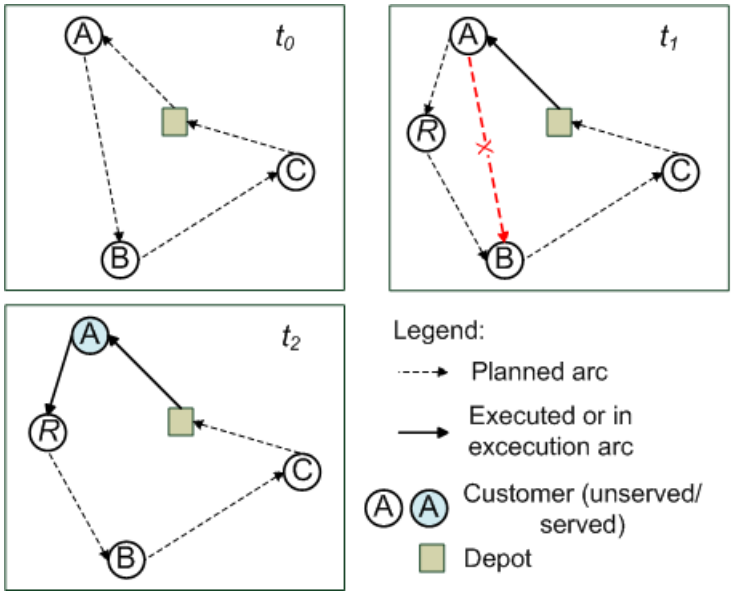

Figure 2.1: *Example of ongoing planning.*

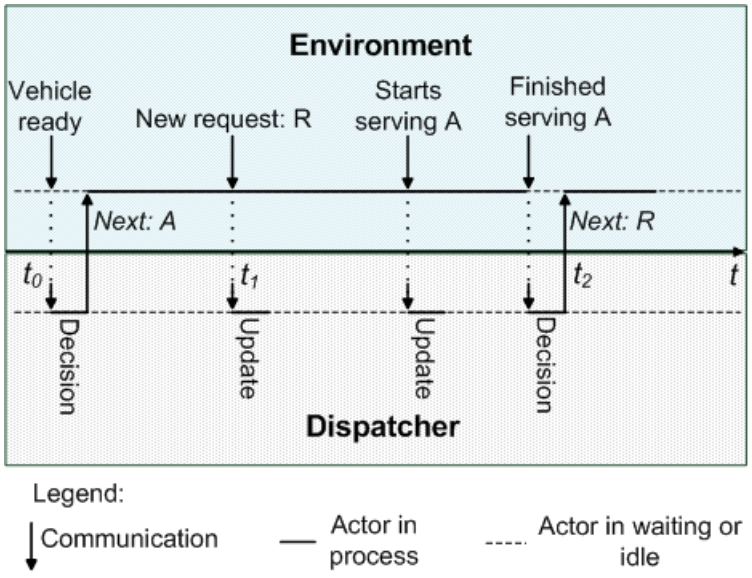

Figure 2.2: *Timeline of communications.*

mands [16], [17], [18] or the cancellation of orders [19]. Moreover, different causes of dynamism are receiving attention such as travel times [20], [21], [22], service times at customers [22], and breakdown of vehicles [15]. In Table 2.1 the reviewed articles are classified according to the source of dynamism and its quality of information. References in bold represent papers that consider more than one source of dynamism.

Table 2.1: *Sources of dynamism*

Source of dynamism	Quality of information	References
new requests (new customers)	deterministic	[23] [24] [14] [25] [26] **[19]** [27] [28] [29] [30] **[15]** [31] [32] [33] [34]
	stochastic	[35] [10] [36] [37] **[18]** [38] [39]
demand variation (same customer)	deterministic	**[16]** **[17]** [40]
	cyclic	**[17]**
	stochastic	**[18]**
order cancellation	deterministic	**[19]**
request times	deterministic	**[16]**
service time	stochastic	**[22]**
travel time	deterministic	**[16]** **[15]** [20] [21]
	stochastic	**[22]**
breakdown	deterministic	**[15]**

2.2.2 Challenges of the DVRP

Time dimension is essential for DVRP [4]. On one hand, real-time capabilities/technologies make possible more realistic way to face real-world problems (produce solutions closer to real scenarios). On the other hand, faster decisions and responses are required. Availability of real-time information opens the door to a series of features and problems that put the scheduler in front of a harder decision process, with more variables and less time to work out a solution. For example, in the context of ongoing arrival of requests, a first new problem arises: should the courier accept or deny the customer's request [3]?

As a consequence, a measure of accepting rate should be taken into account while judging the worth of the solution in the final balance. Again, most customers call for an *as soon as possible* service, so minimizing the *response time*, i.e., the delay between the arrival of a request and its service should be the main objective.

Therefore, whereas in static VRP the objective function deals with minimizing some sorts of travel costs (distances, times, emissions, etc. [41]), in DVRP the objective function may introduce other notions such as throughput (number of customers serviced per unit time), service denials, response time, revenue maximization [3], [14] and customer satisfaction [36].

Another new aspect is the introduction of *diversions* into the DVRP [35], [15], [10]. According to Ferrucci *et al.* [10], vehicle en-route diversion describes a situation where a vehicle is currently traveling to a request location when another request, which is to be serviced next, is assigned to the vehicle by a dispatching center. In most papers, diversions are not allowed: once a driver has been notified about his next customer, it can not be changed [14]. However, as revealed in [35] and [10], satisfying customers with urgent requests might require diversions, but their number should be taken under control (by penalties applied to less convenient diversions) in order to avoid negative consequences due to excessive distractions for drivers (off-road glance to navigators) and drivers' discontent [10].

In addition, response time imposes a compromise between reactivity and decision quality. For this reasons heuristics, meta-heuristics and ILP-hybridization are preferred to pure exact methods, while some forms of decentralization [38], [20] have been introduced.

Finally, complexity of the problem, reactivity and ongoing fashion, may require some form of help to decision maker, so DVRP solvers are often part of Decision Support Systems [11], [19], [16].

Hereinafter, unless otherwise indicated, the source of dynamism will be the arrival of new requests.

2.2.3 Degree of dynamism

The Degree of Dynamism (DoD) is the measure of how many events are received after the initial planning. The first and most common definition is due to Larsen *et al.* [42] and represents the ratio between the number n_d of ongoing new requests and the total number n_{tot} of requests at the end of the transportation process:

$$DoD = \frac{n_d}{n_{tot}} \tag{2.1}$$

More complex formulations are reported in [3], taking in account also the lateness.
As a novelty, Lin *et al.* [19] introduce a variation considers dynamic cancellation of orders or requests:

$$DoD = \frac{n_{cancel}}{n_{off}} \frac{n_{new}}{n_{off} - n_{cancel}} \tag{2.2}$$

where n_{cancel} is the number of canceled customer orders, n_{off} is the number of known customer orders (offline), and n_{new} is the number of new customer orders.

2.3 Applications

DVRP assumes a continuous communication between the dispatcher and the vehicles (or among logistics entities), in order to monitor execution of routes and implement real-time control strategies, i.e., re-optimization of the routing plan.
Technologies makes it possible to solve the dynamic routing problems for a wide range of real world applications, where just-in-time reaction may be crucial.
This section reports a set of recent contributions classified on the basis of application fields.

2.3.1 Services

In this class of applications, vehicles have to visit customers and supply them with a sort of service, generally with time windows.
Typical service domains fall into the area of maintenance and repair operations [22]. A traditional formulation is the Traveling Repairman Problem (TRP) with the objective of

minimizing waiting time and service time for customers, rather than the overall traveled distance as in Traveling Salesman Problem (TSP) [4]. Objectives of TRP were already close to DVRP and recent papers about service application in a dynamic context are reported in [22] [36].
In addition, [36] considers contexts where the customer satisfaction is accomplished after more than one visit in the same planning horizon. Such applications spread across large scale disaster management, medical diagnosis and health care, forest fire handling, service vehicle repair and real-time tactical reconnaissance military missions.

2.3.2 Transport of goods

About transport of goods, the most challenging environment for DVRP is city logistics [3], [41], where traffic congestion and other events are more likely to disrupt routing plans. Furthermore, customers often require "same day delivery", calling the system for shorter reaction time to changes in customer demands and traffic conditions [41].
In order to support decision makers of a hypothetical carrier agency, in such a complex and time-critical environment, [11] integrates DVRP into a DSS architecture, together with modules for acquiring real-time traffic data and forecasting travel times on the basis of real-time and historical data.
In the typical urban scenario, transportation is in charge to couriers (carriers), who serve shippers and end-customers (citizens) with deliveries, pickups or both in the same planning horizon (day). City logistics tries to globally optimize costs and benefits for different stakeholders [41], so some of the considered papers face multi-objective optimization [23], [24], [26]. The aim is balancing customer satisfaction with travel costs, in charge to carriers.
A holistic objective should take in account citizens' living conditions [41], which are affected by pollution and traffic congestion. Nevertheless, in the reviewed papers, pollution is not explicitly considered, although DVRPs facing dynamic travel times and congestion could be seen as a general way to reduce emissions. However, readers interested in green logistics can find information about Green VRP (GVRP) and Pollution Routing Problem (PRP) in [2]. In few words, GVRP introduces fuel consumption and involves also

alternative-fuel powered vehicles, whereas PRP considers CO_2 emission models. Apart from the traveled distance, both consumption and emission consider speed, effective load and engine status of a vehicle.

In some papers, assuming that accomplishing all the customers may be impossible (time windows), priority of orders (requests) is considered. Priority may depend on how long a pending order has been waiting for service, hence postponed orders assume maximum priority for the day after planning [35], [24], [26], [27].

In the field of waste collection services modeled as DVRP, paper [40] points out that uncertainty about filling levels of bins (uncertainty about demands) may cause unexpected early saturation of vehicles. In this case, all other traveling vehicles are requested to wait until an Ant Colony Optimization (ACO) algorithm decides either to employ a new vehicle or reroute an already traveling one.

Finally, as an example of not packaged goods, paper [30] concerns the optimization of the journeys of refueling trucks in an airport. The trucks must deliver fuel to aircrafts during predefined time windows corresponding to a sub period of the time on the ground. In this case neither delays nor discarded request are allowed, so responsiveness may call for auxiliary vehicles.

2.3.3 Transport of people

Transport of people is quite similar to VRP with pickup and delivery [39] and has been often modeled as a Dial-a-Ride Problem (DARP), where main emphasis is given to customers satisfaction and reduction of passenger inconvenience [37].

In a typical scenario, buses (or vehicles in general) have initial routes/lines, but customers can call the dispatcher in order to notify late requests. Every late request specifies boarding and alighting nodes, a time windows, a service duration and a number of persons to be transported [37]. The dispatcher must decide if to accept or deny a late request and then which vehicle should be deviated from the planned route.

Munoz *et al.* [39] faces a more dynamic environment where all requests are revealed in real time, but a continuous re-optimization process exploits demand forecasting in order to route vehicles step by step.

An earlier article by Coslovich *et al.* [43] showed a decentralized approach in managing dynamic requests. In their case study, at a regular bus stop and before getting on, an unexpected customer can ask directly the driver for an unplanned alighting stop. On-board continuous and autonomous re-optimization process supports the driver in making fast decisions about accepting or denying the request. Usually, a late request is accepted if the caused detour does not decrease service level for early (planned) customers under a certain threshold.

2.3.4 Others

Due to green regulations and further technological progress in the field of Electric Vehicles (EVs), green logistics [2] is assuming more interest. The challenge is incorporating energy based restrictions into vehicles routing problems. For instance, in the aforementioned GVRP, vehicles are allowed to recharge during their tours to extend the distance they can travel [2]. One of these restrictions is the limited battery capacity which makes detours to recharging stations necessary. However, at the best of our knowledge, no papers on DVRP explicitly consider EVs.

2.4 Solution Methods

Along with advances in technologies, improvements to problem-solving methods have been made in order to interact with their dynamic environment and to combine fast responses with solution quality [14].
In this section, recent papers are reported by following the classification proposed in [3], and also distributed approaches are considered. The mentioned solution methods with the specification of the related approaches are summarized in Fig. 2.3.

2.4.1 Dynamic-deterministic routing problem

Dynamic-deterministic routing problems are solved in two main centralized approaches: a) in *periodic reoptimization*, DVRP is faced as a series of static and independent VRP instances, each of them holds in a time interval of the workday (or execution horizon);

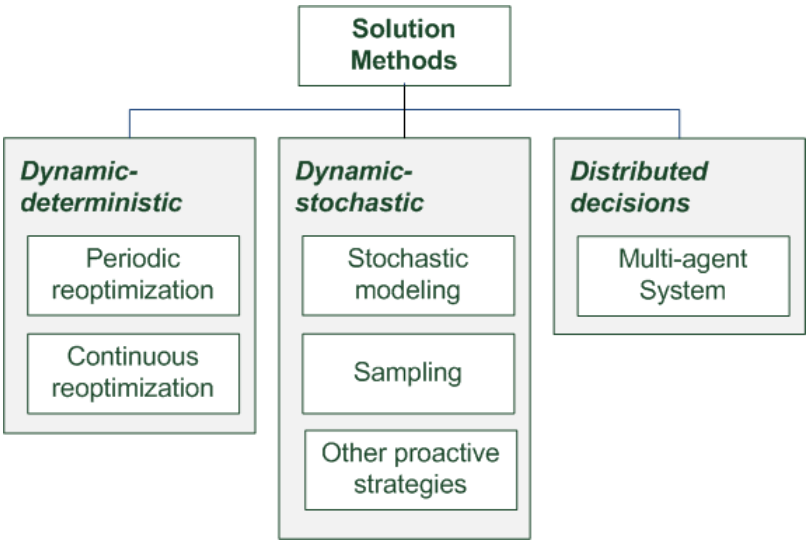

Figure 2.3: *Solution Methods.*

b) in *continuous reoptimization*, an adaptive memory, most of the time a pool of good solutions, is continuously updated and utilized as a feature to decide only the next destination of each vehicle [14].

Periodic reoptimization

Periodic reoptimization approaches face DVRP as a series of consecutive static VRPs, each of them holds only in a fraction of the whole time horizon. Fractions can be of fixed duration or event-bounded. The typical periodic reoptimization approach gives a first optimal initial routing plan (set of routes) before the beginning of the workday. After the transportation has begun, an optimization procedure periodically solves a static problem corresponding to the current state, either whenever the available data change [26], [25], or at fixed intervals of time, called decision epochs [33] or time slices [24].

For example, [24] and [23] divide the planning horizon into time slices of duration Δ, compatible with the *DoD*. At the beginning t_i of a time slice $[t_i, t_{i+1}]$ (where $t_{i+1} = t_i + \Delta$), a certain amount of time δ is spent to organize the required information and then to construct the static model. This model is solved in $[t_i + \delta, t_{i+1}]$ to find the solution S_{i+1}, which should be implemented (executed by vehicles) in the next time slice $[t_{i+1}, t_{i+2}]$. Each single VRP is solved by a Genetic Algorithm (GA).

Sarasola *et al.* [18], for example, divide the day into 25 periods and solve a static VRP in the first 5 secs of each period, by a Variable Neighborhood Search (VNS) employing a Clarke and Wright savings algorithm.

The same daytime division in time slice is considered by Okulevicz *et al.* [28] that use a two-phase Particle Swarm Optimization (PSO). In each time slice, a PSO algorithm assigns clients to vehicles, then different PSOs (hybridized with 2-Opt algorithm) solve a TSP instance for each single vehicle. The center of the search space for the particular time slice is chosen based on the best known solution from the previous time slice while the order of requests that were not present before is generated at random.

Following Montemanni *et al.* [44], Euchi *et al.* [31] solve static VRPs by an ACO algorithm in each time slice and with pheromone conservation from one slice to the next.

A different ACO is used in *et al.* [17] where authors consider a particular and artificial scenario thought to demonstrate the effectiveness of a new proposed benchmark. In the considered scenario, customers and global demand are known, whereas single demands are pairwise swapped in order to simulate single demands changing. The variation is modulated by a frequency f and a magnitude m of changes, while the time horizon is divided by the iterations of the algorithms, one for each 'static' environment as it is at the beginning of each iteration t. More precisely, the authors propose a framework of ACO algorithms that maintains a population of the best ants (best VRP solutions) of every iteration, while a percentage of worst solutions is replaced by newly generated immigrant ants. Then, the framework regenerates the pheromone trials considering information only from the previous environment and extra information from the newly generated immigrant ants. By the proposed benchmark, the authors compare results for different ways of generating immigrant ants (immigrant schemes), as f and m change. Generally, generation driven by the last best ant performs better than random generation because the first allows an effective knowledge transfer throughout the time. Even if the application focuses on a particular scenario, the authors make an effort to present an suitable benchmark for DVRP.

The "VRP-serialization" has the advantage of employing well-known algorithms developed for static routing. On the other hand, the drawback is the delays for the dispatcher,

as one can guess by the example [24]: during a time slice, vehicles perform operations decided on the basis of data available at the beginning of the previous time slice. However, in this class of approaches, drivers often know the initial plan and may get annoyed by continuous detours. Therefore dynamic events are handled trying to modify the initial plan the least possible [16], [25], [18].

Continuous reoptimization

In the continuous reoptimization approaches, optimization is continuously performed over the time horizon, while an adaptive memory stores information on good solutions. More precisely, whenever the available data change, a decision procedure aggregates the information from the memory to perform reoptimization. The terminology *Adaptive Memory Programing* (AMP) is explained in [45], where AMP has the aim of grouping metaheuristics that memorize solutions or characteristics of solutions generated during the search process into the solution space. Examples of AMP are solution pools [14], [15] and ACO with pheromone conservation. About ACO, it is worth to notice that, differently from ACO reported into subsection 2.4.1, the strategy restarts the optimizer each time new information is available [30].

Barkaoui and Gendrau [14] propose a significant example of approach based on a solution pool. The authors consider a set of vehicles that are performing a set of planned routes which compose the best current solution. Each route begins with the best current destination of the associated vehicle and the best solution is extracted from a solution pool that a GA continuously improves by a search process into the solution space. The solution pool has to be consistent with the current environment, i.e., all the solutions must begin with the same current destinations for each vehicle. In order to maintain consistency of solutions, the search process is interrupted whenever a triggering event (an input update) occurs. Then, firstly solutions are properly updated, secondly the search process can restart. More precisely, two types of events are considered: i) if a vehicle has finished serving its current customer (previous current destination), the new best solution is extracted by the pool and used for determining the next destination, while the remaining solutions are consistently updated; ii) when a new service request

arrives, it can be accepted and included in all the solutions or rejected if there is no feasible insertion position in any solution.

Ferrucci *et al.* [15] follow the approach of solution pool by introducing the notion of concurrency of execution and adaptation along the execution time, which is divided into short *anticipation horizons*. In each anticipation horizon the best solution, known as relevant plan, guides vehicles movements, while theoretical plans are concurrently adapted according to the relevant one and re-optimized. At the same time, every dynamic event is buffered and considered at the beginning of the next horizon. All the plans are recomputed by Tabu Search (TS) and the best one is chosen as the new relevant plan.

Schyns [30] proposes an adaptation of ACO in a context where the objective is the responsiveness, i.e., servicing a customer as soon as possible in its time window. The strategy is to restart the optimizer each time new information is available, together with pheromone conservation from one optimization to the next one, in the dynamic context.

The main advantage of this class of approaches is that decisions are taken on the basis of a set of good solutions that can be easily adapted to the current situation and are suitable for parallelization. The main drawback is that only reactive strategies can be adopted because no knowledge about the future is available, such as the time-space distribution of future requests. In addition, if responsiveness is not so strict, retaining new service requests in a buffer could also be considered. By collecting a batch of requests and by dispatching them all at once, the adverse effects of dispatching requests in a sequential fashion could be alleviated [46]. About such a remark, a possible answer could be *anticipation horizons* proposed in [15].

2.4.2 Dynamic-stochastic routing problem

In dynamic-stochastic routing problems, an additional stochastic knowledge about the source of dynamism can be exploited. Pillac *et al.* [3] recognize two main categories of centralized approaches: a) in *stochastic modeling* approaches, the additional knowledge is formally included into the problem formulation; b) in *sampling* approaches, several realizations of random variable distributions (e.g. late customers requests) are performed in order to obtain the most likely scenarios in the near future and anticipate the source

of dynamism.

Stochastic modeling approaches

Binart *et al.* [22] tackle a service on fields routing problem where the source of dynamism are travel times and service times (minimum and maximum values are known), whereas customers (mandatory and optional) are known. The proposed method is composed of a planning stage and an execution stage. In the planning stage a linear program is solved in order to find out routes visiting all the customers (mandatory and optional), under pessimistic conditions (maximum values for travel times and service times). Pessimistic values are likely to lead to unfeasible routes (planning horizon violation), so a repair heuristic procedure removes the farthest (from segments between mandatory customers) optional customers until the feasibility. In the execution stage, dynamic programming is applied each time a vehicle finishes servicing a customer, in order to decide if to send it to the next mandatory customer or to an optional customer (even those discarded by the repair procedure).

Muñunz-Carpintero *et al.* [39] propose a Hybrid Predictive Control approach to control a dial-a-ride system. In the proposed predictive dynamic model, the objective function depends on both current requests and expected future requests in a certain prediction horizon. The optimization problem is solved using an evolutionary algorithm each time a new request arrives, giving routes that hold only until the next new request arrival. GA and PSO are considered and compared with each other.

The merit of this class of approaches is the attempt to introduce uncertainty into the model and hybridize exact methods with heuristics for reasonable computation time. However, the main drawback remains the required computational effort. In fact, the solution proposed in [22] performs effectively on instances containing up to 50 customers and 3 vehicles, while [39] considers 8 requests during a time horizon of two hours.

Sampling approaches

According to Bent *et al.* [47], the key idea behind this type of approaches, which is an enhancement of the continuos reoptimization for dynamic deterministic problems, is to continuously generate and solve scenarios which include both known and dynamic requests. Each scenario is solved into a plan which is then *projected* on the known requests. Decisions during plan execution are based on a distinguished plan which evolves overtime and is updated on the basis of scenario-specific plans. In [47], the distinguished plan is selected by a consensus function that chooses the plan most similar to the current pool. Time between decision is spent to continuously improve the current scenario pool. Sarasola *et al.* [18] face the problem of VRP with stochastic demand and dynamic requests. The particularity is that even static customers (known before the planning horizon) have stochastic demands: that is, customer demand remains uncertain until a vehicle arrives at the customer location. At first, authors use sampling to extend a VNS algorithm to static-stochastic problems. The new algorithm, denoted by Stochastic-VNS (S-VNS), is then adapted to dynamic settings (dynamic-stochastic problems) with time horizons divided into regular periods. This new adaptation is called Stocastic Dynamic VNS (SD-VNS). The dynamic-stochastic DVRP is solved as period-specific sampled VRPs. More precisely, a mean on a high number of realizations, for each period, reduces the problem to a series of distinct deterministic VRPs, one for each of 25 periods of a same day. A stochastic event scheduler gives late customers positions and, for all customers, the real demand when visited. As authors point out, the proposed SD-VNS is good for cases with low degree of dynamism (5%-15%) and all dynamic customers revealed in the first half of the time horizon. The main aim is not only to determine the next destination of a vehicle, but to give VRP solutions employing typical static-stochastic recourse action, such as the detour-to-depot when capacity constraint is violated.

Bruni *et al.* [37] address the robust demand-responsive planning problem under uncertainty of customers by formulating it as a stochastic variant of the DARP, in order to include recourse actions (detours). By sampling the stochastic distribution of not yet revealed customers, the proposed procedure generates different scenarios, named sub-

problems, involving both the known customers and scenario-specific customers. In the first stage, every sub-problem is solved by TS. In the second stage, a coordination mechanism merges scenario solutions by evaluating a series of similarity measures. In this way, a unique robust solution is found, which combines pro-activity allowed by stochastic knowledge with robustness given by the coordination mechanism.

Barkaoui *et al.* [36] tackle a generic problem in which a customer may require more visits before its satisfaction is eventually reached (multi-step repairs, visits for a diagnosis, etc.). They model this behavior by assigning to each customer a satisfaction value which is increased according to a probabilistic function, visit by visit, until a threshold is attained (customer is considered fully served). In this way, they build a stochastic knowledge about customers that are likely to require new visits in the near future. This knowledge is exploited into their solving strategy: a hybrid GA continuously generates plans including both revealed requests and potential future revisit requests (*dummy* revisits). If many visits are candidates for insertion in planned routes, then the first visit to insert is the one with the minimum deviation of the customer average satisfaction from the satisfaction threshold. The average is computed by simulating (sampling) plan execution over all possible histories (sequence of events) for a given solution plan.

This class of approaches has the merit of attempting to exploit knowledge about future to improve the decision process. However, the experiments reported into the cited papers concern uncertainty on at most one source of dynamism a time, i.e., only one stochastic variable is considered. Moreover, time for decisions could be no longer enough for a centralized dispatcher as the dimension of the problem grows.

Other strategies

Stochastic knowledge about the evolution of the source of dynamism (e.g. information about spatial and/or time distribution of late requests) allows other pro-active approaches which anticipate future events (e.g. requests).

Ferrucci *et al.* [35], [10] integrate stochastic knowledge by adding so-called *dummy customers* to the set of real pending customer requests. Dummy customers guide vehicles to request-likely areas before real requests occur there. Stochastic knowledge is generated

from past requests.
Request-likely areas are the base principle also for waiting strategies, in which it needs to decide when, how long and where the vehicle should wait. A vehicle can wait at the last serviced customer, or can go to a strategical parking area for a waiting period [9].
Albareda-Sambola *et al.* [27] face a problem in which a rolling horizon is divided into sub-periods and customers have time windows covering more sub-periods. An adaptive policy exploits stochastic knowledge about next sub-period arriving requests (non-pending customers) in order to postpone some currently known customers (pending customers) to the next sub-period. The criterion is a geographically-based *compatibility index* between pending stochastic customers. Then a VRP on the remaining pending customers in the current sub-period is exactly solved or, for large instances, VNS is used.

2.4.3 Distributed approaches

The challenge of giving fast responses to frequent unexpected events, into large transportation and logistics networks, has made some researchers investigate on a paradigm shift from a centralized approach towards decentralized control of logistic processes, in order to avoid too time consuming re-computations of global routing plans [34]. This shift has been usually made by Multi-Agent Systems (MAS), in which autonomously acting software agents represent logistic entities (carrier, vehicles, orders, etc.). Agents have the ability to interact with other agents, by the use of negotiation and communication mechanisms [32], and to take decisions.
Although different agent hierarchies can be found in the literature, usually the main agents are those acting on behalf of orders and vehicles [32], [34].
Warden and Wojtusiak [34] consider a freight forwarding agency whose transportation process is modeled by two main classes of agents: transportation orders and trucks. More precisely, each time an incoming transport order arrives, a short-lived agent is created and becomes responsible for the re-valuation of the pending order, until it is included into a route or discarded. In particular, the longer an order remains pending, the more its priority. Meanwhile, each time a truck agent reaches a storage facility, its agent performs two tasks in order to choose the next destination: (a) order selection; (b) optimize

Table 2.2: *Solution Approaches*

Type of problem	Approach	Solution	References
Dynamic-deterministic	Periodic reoptimization	VNS	[19] **[18]** [25] [26]
		GA	[23] [24]
		TS	[33]
		PSO	[28] [29]
		ACO	[31] [17] [40]
		AMP	[16]
	Continuous reoptimization	GA	[14]
		TS	[15]
		ACO	[30]
	Distributed	MAS/auction	[20] [21] [32] [34]
Dynamic-stochastic	Stoch. modeling	B&C or Lagr.	[22]
		PSO or GA	[39]
	Sampling	VNS	**[18]**
		TS	[37]
		GA	[36]
	Others	TS	[35] [10]
		exact or VNS	[27]
	Distributed	MAS/auction	[38]

transportation plan. The first task produces a solution skeleton as a list of n next destinations, chosen and sorted by maximizing a weighted sum of gains (values of orders) and costs (travel costs). The second task transforms the skeleton into a transportation plan by a learnable evolution model, which is an evolutionary optimization method that employs machine learning module to direct the evolutionary process. A central order repository agent avoids conflicts between trucks. However, authors themselves declare the need of a more explicit modeling of the outer boundary of the forwarding agency (for instance, real contract negotiation with customers).

Negotiation and a system-wide approach are proposed in [38], where a MAS based on interactions between shipper and carry agents improves look-ahead strategies for pickup and delivery jobs. The aims are minimizing the system-wide costs (transport costs and lateness costs) and avoiding allocations which may become unfavorable when new jobs appear. In other words, the authors study the emergent behavior of the combined system shippers-carries by evaluating *joint* effects of their policies (auction strategies vs bidding strategies, respectively). The most interesting policy the authors propose is the Opportunity Valuation (OV) policy, applied by vehicles/carriers agents each time a shipper agent starts an auction for a new job. On the basis of stochastic information on job arrival patterns and distribution of the lowest bid for various job characteristics, the OV policy (at each vehicle) employs Stochastic Dynamic Programming in order to: (i) calculate a bid price for the new job, (ii) choose an appropriate insertion position for the new job, and (iii) support so-called pro-active move decisions, i.e, moving empty in anticipation of future job requests.

Novaes *et al.* [20] consider traffic congestion as source of dynamism in a problem involving a plant, acting as depot, and Original Equipment Manufacturers (OEMs) acting as nodes to be visited. The used MAS involves a depot agent and vehicle agents. Each vehicle of a fleet tries to perform an initial route but unpredictable traffic congestion can give faults, i.e., some OEMs are no longer reachable with respect to working time constraints. In order to anticipate the occurrence of unperformed tasks (the visits), each time a vehicle has finished servicing an OEM, its agent performs a sequential analysis on the speed [48]. The aim of the analysis is to consider only persistent slowdowns as early indications

of over-congested traffic conditions. Therefore, the analysis relies on an experiment composed by observations on the last speed and a certain number of previous speeds along the path up to the OEM. The experiment is associated to a value that is compared with a normal scenario and a congested scenario. Such scenarios are built on the basis of historical and typical traffic patterns and are characterized by a normal speed and congested speed, respectively. Moreover, these scenarios are discriminate by a lower and an upper threshold that take in account the probability of confusing a scenario with the other one. Therefore, assuming a vehicle is in the normal scenario, at a certain stage the value of the experiment is compared with the thresholds. If the result falls between the thresholds, the decision is postponed, i.e., the experiment continues by waiting the speed at the next stage (next observation). Otherwise, the agent can make a decision and the experiment can restart. More precisely, if the result goes under the lower threshold, an incipient traffic congestion is detected and the congested scenario is assumed hereinafter. Then, the vehicle agent considers congestion speed and elapsed time along the route in order to estimate the portion of initial route it can actually travel, and it tries to transfer non-executable tasks to other agents. If no agreement is reached, the depot agent sends an auxiliary vehicle. However, as the authors state, [20] focuses mostly on the fault detection by a sequential analysis, leaving the transference of tasks as an open problem (which task are to be transferred, negotiation among vehicles, etc.).

In a more recent paper [21], the authors focus on the transference of tasks, into a slightly different problem: OEMs become suppliers, while the depot becomes an OEM; a hierarchy of agents is introduced. In particular, once some vehicles detect incipient faults, a Vickrey auction [49] is adopted in order to negotiate the transference of tasks to other "bidder" vehicles, under the control of a unique carrier agent. Auctions take place at fixed intervals set by the carrier agent, which is also responsible of sending auxiliary vehicles if the agreement is not reached. Authors choose Vickery auction, also known as *second-price sealed-bid auction*, to assign the auctioned item (the task in this case) to the best bidder. The price is the one submitted by the second best bidder, in order to guarantee bids that reflect the real and fair value of the item. Therefore, this method is particularly suitable for collaborative contexts in which there is no incentive

for any bidder to misrepresent the value of the item. Moreover, paper [21] presents a review about negotiation mechanisms in MAS and an idea for a web based architecture for production-transportation integrated problems. Simulations require a MAS platform such as PlaSMA [34], [32] or Plant Simulation [38].

Finally, the mentioned papers are listed in Table 2.2, where they are classified on the basis of approaches and type of solutions. More precisely, the following notations are used: i) B&C and Lagr. are branch and cut and Lagrangian relaxation respectively; ii) bold references denote papers covering more than one approach, iii) AMP stands for Adaptive Memory Programming not involving well known metaheuristics such as TS or GA.

2.4.4 Benchmarks

It is enlightened in [3], [14] and [41], that specific benchmarks for dynamic VRP are not available. Nevertheless, some contributions such as [14] test their proposals using some adaptations of Solomon benchmark by Gendreau *et al.* [46], in which a fraction of the customers is revealed dynamically. An example is shown in [47].

Schyns considers, in [30], some classical VRP benchmarks with extensions to the responsiveness context.

Mavrovouniotis *et al.* [17] observe that the main issue with different dynamic benchmark generators is that the optimum value is not known during the dynamic changes, and this makes impossible to answer to "how close to the moving optimum an algorithm performs when a change occurs?". For the particular and artificial case of a DVRP in which the number of customers and the overall demand do not change over time, the authors propose their Dynamic Benchmark Generator for Permutation problems (DBGP). The DBGP builds an artificial DVRP problem as a sequence of static VRPs instances obtained by periodical pairwise-swap of demands between customers. In such a particular situation, the authors state that, with the DBGP, the optimum value remains the same over environmental changes and, thus, one can see how close to the optimum an algorithm converges as well as comparing the performance between different algorithms.

Chapter 3

A VRP-based DSS for urban services

3.1 Introduction

Urban logistics services are complex and socially, environmentally and economically sensitive: they require dedicated and intelligent support from the decision making process point of view.

In the context of city logistics, Postal Delivery (PD) and Waste Collection (WC) services are considered core problems [50] and they are often modeled as VRPs. Therefore, on one hand, there is a continuous effort in researching effective and efficient solving methods, able to face different and increasingly complex variants of VRP. On the other hand, research can bring tangible benefits if its results are exploited into ICT systems, whose increasingly availability, in turn, allows the development of models and novel decision making strategies to optimize the information process and utilization for the Decision Makers (DMs).

For instance, in the field of the waste collection routing problem (WCRP) the are several approaches and strategies in order to deal with complexity and dimensions of real systems [50]: i) genetic algorithms [51]; ii) heuristic algorithms based on arc or node routing problems [52], iii) swarm intelligence methods [53] [54] and ants heuristics [55]; iv) simulation approaches [56].

In order to provide contributions in context of city logistics, this chapter first provides a brief description of an architecture for a DSS devoted to services like postal delivery and waste collection. Then, the chapter focuses on the the core of the proposed U-DSS: the VRP module that provides routing plans with respect to time and capacity constraints [57].

It is well known that these kinds of problems are NP-hard and can be solved in reasonable time only if the dimension of the system is quite limited. Hence, we solve the considered VRPs by a two-phase heuristic algorithm: the first phase is based on a clustering strategy and the second phase is based on a farthest insertion heuristic for the solution of the TSP [58].

The proposed general algorithm is specified for the PD and WC services. Moreover, in this chapter we present the Mixed Integer Linear Programming (MILP) models of the two services: comparing the algorithm solutions with the optimal solutions provided by the MILP, we show that the heuristics gives good solutions and can be used to solve large and complex problems in short time.

In order to show the applicability of the proposed U-DSS, we consider the Apulian (Italy) PD service and the WC service of the city of Trieste (Italy).

The chapter is organized as follows. Section 3.2 introduces the U-DSS architecture and Section 3.3 describes in detail the VRP module of the model component by proposing a general heuristic algorithm. Then, Sections 3.4 and 3.5 present two instances and customized applications of the VRP module for the PD and WC services, respectively, and two case studies assess the proposed algorithm.

3.2 Urban-Decision Support System Architecture

In this section the U-DSS architecture and its main components are briefly described. The U-DSS is developed following user centered [59] and model based [60] approaches: the first activity consists on the identification of the user needs and requirements through a series of interviews; then the second activity consists on building the U-DSS modules giving emphasis on mathematical models and optimization techniques.

The structure of the proposed U-DSS consists of three main components: the Data Component (DC), the Model Component (MC) and the Interface Component (IC). Fig. 3.1 shows a simple scheme of the U-DSS architecture by enlightening the connections among them.

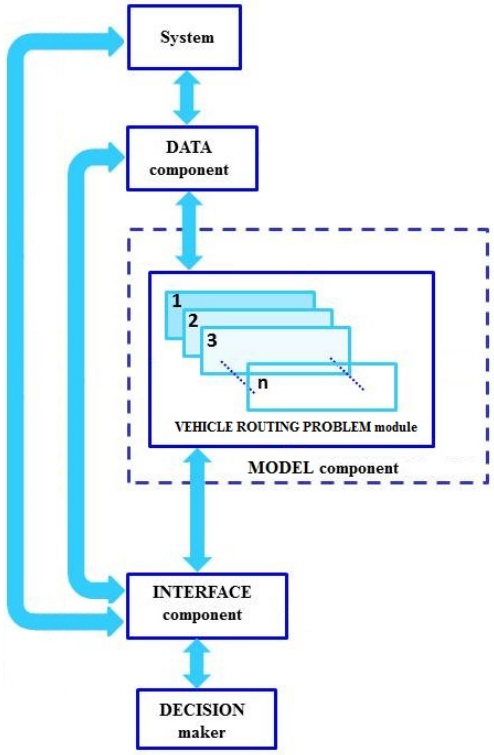

Figure 3.1: *The scheme of the U-DSS architecture.*

The DC collects the data and the information necessary to the U-DSS services: historical data and real time data. In the cases of the VRPs, the historical data base collects the sites to be served, the demand of every site, the number and the dimension of the vehicles, the time length of the shift.

Combining historical data with real-time data makes it possible to refine inputs for next decisions or select inputs that better fit with a particular scenario. Real-time information, for instance travel times along routes detected by GPS, can be stored in the DC in order to update the historical data and forecast more suitable system parameters for different scenarios.

On the other hand, the MC is the core of the U-DSS and contains models, algorithms

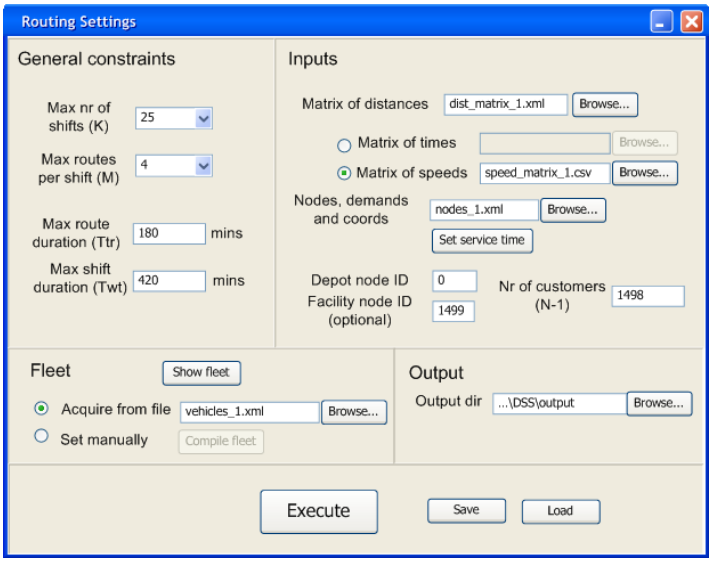

Figure 3.2: *IC input form.*

Shift	Route	Route duration (mins)	Route length (km)	Load
1	1	173,57	47,85	33
	2	179,54	53,49	35
2	1	179,23	54,49	44
	2	173,21	41,81	55
3	1	174,53	53,78	49
	2	176,76	47,60	60
4	1	176,86	36,97	86

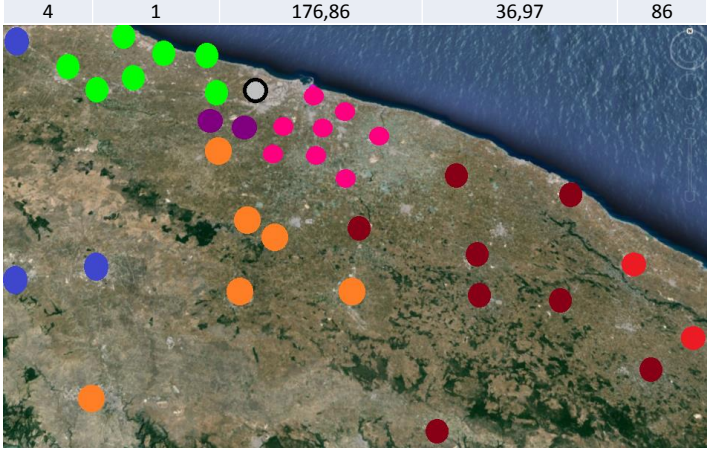

Figure 3.3: *IC solution display.*

and rules needed to provide decision support for the users. In particular, the MC is composed by a module dedicated to solve VRPs. In the next section, the module is described in detail by proposing a general heuristic algorithm that can be customized to be applied to different services based on vehicle routing problems.
Finally, the IC allows interaction between users and the real system: it is responsible of the communication and interaction among the MC, the DC, the real-world and the users. The IC main tasks are: i) setting the information needed for the MC to operate and determine the optimal solution; ii) displaying the solutions provided by the MC; iii) connecting the U-DSS with the real-time data sources. For instance, Fig. 3.2 shows the panels that the user has to fill in order to provide the necessary information and Fig. 3.3 shows some possible U-DSS text and graphic outputs of the IC.

3.3 Vehicle Routing Problem Module

In this section the MC of the proposed U-DSS is described: it considers the general problem of optimizing the routes of a set of vehicles that have to visit a set of customers by minimizing the total traveled distance, under capacity and time constraints. Our effort is focused on presenting a general model that can provide solutions for different urban logistics problems: suitable instances and customized applications of the proposed module can be realized in the MC.

3.3.1 Problem Description

Let consider a set of vehicles $\mathcal{U}$ that operate collection or delivery activities subject to time and capacity constraints. During a work shift, each vehicle can follow several routes that may be limited by a maximum of M routes per shift. In order to associate a shift to a vehicle, we simply denote by k the shift performed by vehicle $u_k \in \mathcal{U}$ of capacity Q^k. In addition, each route starts from a depot, follows the assigned path and, when the vehicle attains capacity or time limits, it goes to an intermediate or final destination that can coincide with the depot. In this site the workers can perform an operation and/or make a stop before starting a new route. Each route time has to be less than or equal to

T_r time units and the shift, composed of a set of routes performed by the same vehicle, is limited by the working time of T_w time units.

Let $G = (\mathcal{N}, \mathcal{E})$ be a directed fully connected graph, where $\mathcal{N} = \{v_0, v_1, \dots v_N\}$ is the set of nodes and $\mathcal{E}$ is the set of arcs. In particular, node v_0 is the depot and node v_N represents a facility, that can be a node with specific role such as a landfill in waste collection system or the depot itself in typical VRPs. Moreover, each node v_i with $1 \leq i \leq N-1$ is a customer characterized by a demand $q_i > 0$ and a service time $p_i > 0$. Then, by introducing $C = N - 1$, the set $\overline{\mathcal{C}} = \{v_1, v_2, \dots v_C\}$ denotes the customers set. In addition, each arc $e_{ij} \in \mathcal{E}$ corresponds to the shortest path from v_i to v_j of length c_{ij} and t_{ij}^k is the time spent by vehicle u_k to perform the path of length c_{ij}.

The m-th route of the k-th shift is described by the following elements:

- set $\mathcal{C}_m^k \subseteq \overline{\mathcal{C}}$ contains the customers visited by the m-th route of the k-th shift;
- function $\mathcal{R}_m^k$: $\{1, 2, \dots, |\mathcal{C}_m^k| + 2\} \rightarrow \mathcal{C}_m^k \bigcup \{v_0, v_N\}$, where $\mathcal{R}_m^k(j) = v_i$ means that the j-th node visited during the k-th shift in route m is $v_i \in \mathcal{C}_m^k \bigcup \{v_0, v_N\}$, for $j = 1, \dots, |\mathcal{C}_m^k| + 2$. Note that $\mathcal{R}_m^k(2) \in \overline{\mathcal{C}}$ denotes the first node of route m visited after v_0 or v_N, and $\mathcal{R}_m^k(1) \in \{v_0, v_N\}$, $\mathcal{R}_m^k(|\mathcal{C}_m^k| + 2) \in \{v_0, v_N\}$ are the first and the last nodes of the route, respectively.

Symbol $|(.)|$ denotes the cardinality of set $(.)$.

Moreover, we denote $\bar{m} \geq 2$ the last route in the shift and $\bar{j}$ the total number of nodes visited in the m-th route (i.e., $\bar{j} = |\mathcal{C}_m^k| + 2$).

Now, the overall duration of the route described by $\mathcal{R}_m^k$ is computed as follows:

$$B_m^k = \sum_{j=1}^{\bar{j}-1} (p_{\mathcal{R}_m^k(j)} + t_{\mathcal{R}_m^k(j), \mathcal{R}_m^k(j+1)}^k). \tag{3.1}$$

We remark that B_m^k includes the overall traveling times and the service times of the m-th route in the k-th shift.

In addition, we define $L_{i,m}^k$ the load of vehicle u_k after service at the i-th step of the m-th route:

$$L_{i,m}^k = \sum_{j=1}^{i} q_{\mathcal{R}_m^k(j)}. \tag{3.2}$$

In particular, for $i = \bar{j}$, we define L_m^k as the overall load of vehicle u_k at the end of the m-th route:

$$L_m^k = \sum_{j=1}^{\bar{j}} q_{\mathcal{R}_m^k(j)}. \tag{3.3}$$

At this point it is possible to define the *feasible routes* as follows:

Definition 3.3.1 *We say that route m performed by vehicle u_k is feasible if $L_m^k \leq Q^k$ and $B_m^k \leq T_r$.*

Moreover, the total duration of the routes in a shift must be less than or equal to T_w, i.e.:

$$\hat{B}^k = \sum_{i=1}^{\bar{m}} B_i^k + t_{N0}^k \leq T_w. \tag{3.4}$$

In other words, the duration of the shift takes into account the durations of the shift routes B_i^k and of the final return to the depot t_{N0}^k.
The aim of the VRP consists of designing a set of feasible routes with minimum total length such that:

- $\mathcal{R}_m^k(j) =$

$$= \begin{cases} v_0 & \text{if } m = 1 \text{ and } j = 1 \quad (3.5.a) \\ v_0 & \text{if } m = \bar{m} \text{ and } j = \bar{j} \quad (3.5.b) \\ v_N & \text{if } 1 \leq m < \bar{m} \text{ and } j = \bar{j} \quad (3.5.c) \\ v_N & \text{if } 1 < m \leq \overline{m} \text{ and } j = 1 \quad (3.5.d) \\ v_i \in \overline{\mathcal{C}}, \text{ otherwise} \quad (3.5.\text{e}) \end{cases} \tag{3.5}$$

  for each k;

- $\forall v_i \in \overline{\mathcal{C}}$ $\exists$ only one triple of values (m, k, j) such that $\mathcal{R}_m^k(j) = v_i$, i.e., each node is visited exactly once.

- $\overline{m} \leq M$, i.e., the number of routes performed by each vehicle is less than or equal to M.

More clearly, the two conditions (3.5.a) and (3.5.b) impose that the first route of each shift departs from the depot and the last route of each shift returns to the depot, respectively. On the other hand, the two conditions (3.5.c) and (3.5.d) force each route, but the last one, to end to v_N and each route, but the first one, to start from v_N. In all the other cases the j-th visited node is a customer.

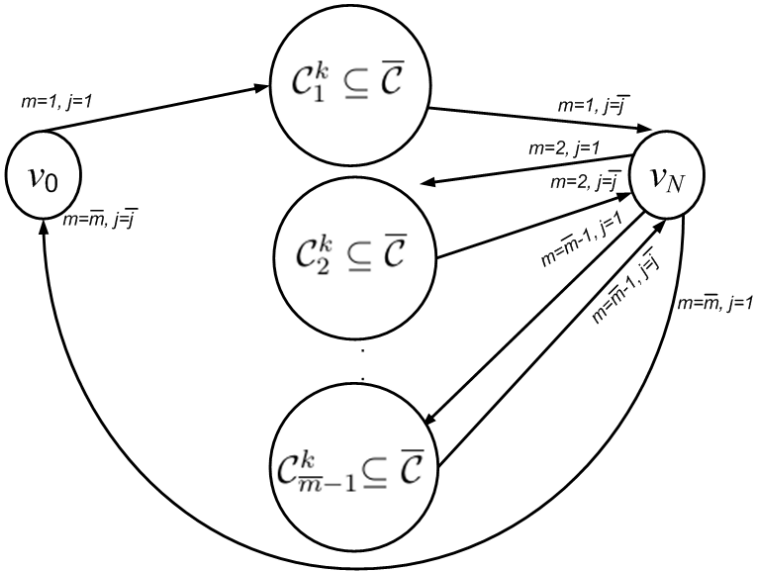

Figure 3.4: *Routes structure.*

Figure 3.4 depicts the described conditions about the routes structure and the necessary notations are summarized as follows:

$\mathcal{U}$ set of vehicles

$\overline{\mathcal{C}}$ set of customers

$C = N - 1$ number of customers

K maximum number of shifts

q_i demand of node $v_i \in \mathcal{N}$

p_i service duration at node $v_i \in \mathcal{N}$

Q^k capacity of vehicle $u_k \in \mathcal{U}$ performing the k-th shift

m the route index in the shift

c_{ij} length of the shortest path from v_i to v_j

t_{ij}^k time to go from v_i to v_j by vehicle u_k

T_r maximum route time of each vehicle

T_w maximum working time of each vehicle

$\mathcal{C}_m^k$ set of nodes visited in route m of the k-th shift

$\mathcal{R}_m^k$ function describing the m-th route of the k-th shift: $\mathcal{R}_m^k(1), \ldots, \mathcal{R}_m^k(\bar{j})$ gives the sequence in which the nodes of $\mathcal{C}_m^k$ are visited

L_m^k total load of cluster $\mathcal{C}_m^k$ defined by (3.3)

B_m^k duration of route $\mathcal{R}_m^k$ defined by (3.1)

$\hat{B}^k$ duration of the k-th shift defined by (3.4).

3.3.2 Vehicle Routing Algorithm

The VRP module of the U-DSS is based on a heuristic algorithm able to solve the problems characterized by the assumptions and conditions described in the previous subsection.

In particular, a two-phase heuristic algorithm is proposed for each shift: the first phase is based on a *clustering strategy* and the second phase is based on a *farthest insertion heuristic* for the solution of the TSP. More precisely, farthest insertion heuristic searches for the not yet inserted node whose distance to any of the inserted nodes is the maximum. Then the selected node is placed into the tour at the point causing the shortest increase of the total length [58].

Let us consider the m-th route of the k-th shift. Before the application of the algorithm, the vehicles $u_k \in \mathcal{U}$ are sorted in descending order of their capacity Q^k.

Phase 1: Clustering strategy. This phase is devoted to build the node set $\mathcal{C}_m^k$ that collects customers positions to be visited in the k-th shift in its m-th route. In this phase the cluster corresponding to $\mathcal{C}_m^k$ is determined by considering only the vehicle capacity constraint: the cluster is built starting from the farthest node from the garage not yet served in the k-th shift (the *seed* node). Successively, the nearest not served node to the seed is added to the cluster and so on. Hence, the cluster is completed when no other node can be added because of the capacity limit of the vehicle u_k.
The basic idea behind the clustering strategy is to consider first the farthest nodes which require a long time to be reached. Indeed, starting from the farthest node in the clusters construction gives a solution in which residual nodes (to be covered by the last clusters) are close among themselves and close to garage, so they can be served with a short route.
Phase 2: Routing strategy. Considering the built cluster $\mathcal{C}_m^k$, the routing is determined by the *farthest insertion heuristic* [58]. If the constraints about the times T_r and T_w are satisfied, then the m-th route of the shift k is completed. Otherwise, the last added customer position in the cluster, which is also the farthest position from the seed, is dropped and the TSP is solved again. Such a procedure is iteratively executed until the constraints on T_r and T_w are satisfied. Then at this point the cluster is closed. If the constraints on T_w or T_r do not permit to create a new cluster (and the related route), then we close the k-th shift.
Such a procedure is iterated for each shift until all the nodes of $\overline{\mathcal{C}}$ are included in one and only one route.

Now, the heuristic algorithm is presented by specifying the steps of the two phases. The following additional sets are introduced:

$\mathcal{U}^U \subseteq \mathcal{U}$ set of used vehicles

$\mathcal{U}^{NU} \subseteq \mathcal{U}$ set of not used vehicles

$\mathcal{C}^S \subseteq \overline{\mathcal{C}}$ set of served nodes

$\mathcal{C}^{NS} \subseteq \overline{\mathcal{C}}$ set of not served nodes

Phase 1. Clustering strategy

Step 0. *Initialization.*

Set $\mathcal{U}^{NU} = \mathcal{U}$, $\mathcal{U}^U = \emptyset$, $\mathcal{C}^S = \emptyset$, $\mathcal{C}^{NS} = \overline{\mathcal{C}}$, $k = 0$.

Step 1. *New shift*

Set $k = k + 1$

If $k \leq K$ **then set** $m = 1$.

else go to Step 9

End

Step 2. *Choosing vehicle*

Select $u_k \in \mathcal{U}^{NU}$ s.t. $Q^k = \max_{u_l \in \mathcal{U}^{NU}} Q^l$.

Step 3. *Determinig the seed of the m-th cluster*

Let $v_s \in \mathcal{C}^{NS}$ be the farthest node from the depot node v_0 not yet included in any cluster (not yet served).

Select $v_s \in \mathcal{C}^{NS} s.t.\ c^k_{0s} = \max_{j:(v_j \in \mathcal{C}^{NS})}\{c^k_{0j}\}$

set $\mathcal{C}^k_m = \{v_s\}$

Step 4. *Populating the cluster with the nearest node to the seed*

Select the nearest not served node $v_n \in \mathcal{C}^{NS}$ to the seed v_s:

Select $v_n \in \mathcal{C}^{NS} \setminus \mathcal{C}^k_m\ s.t.\ c^k_{sn} = \min_{j:(v_j \in \mathcal{C}^{NS})}\{c^k_{sj}\}$

Step 5. *Checking capacity constraints*

If $q_n + L^k_m \leq Q^k$ **then**

set $\mathcal{C}^k_m = \mathcal{C}^k_m \cup \{v_n\}$ and **go to Step 4**.

End if

Phase 2. Solve TSP considering the nodes of the cluster and checking time constraints.

Step 6. *Solving TSP considering the nodes of the cluster*

Set $\bar{j} = |\mathcal{C}_m^k| + 2$,

Select $\mathcal{R}_m^k(1)$ and $\mathcal{R}_m^k(\bar{j})$ according to equations (3.5.a) and (3.5.b).

Determine $\mathcal{R}_m^k(2), \cdots, \mathcal{R}_m^k(\bar{j}-1)$ and B_m^k as solution of the *farthest insertion heuristic* described in [58] and applied to the set $\mathcal{C}_m^k$.

Step 7. *Checking time constraints.*

Determine $\hat{B}^k$

If $\{B_m^k > T_r \vee \hat{B}^k > T_w\}$ **then**

select $v_z \in \mathcal{C}_m^k$ *s.t.* $v_z = \max_{j:(v_j \in \mathcal{C}_m^k)} c_{sj}^k$

set $\mathcal{C}_m^k = \mathcal{C}_m^k - \{v_z\}$

go to Step 6.

End if

Step 8. *Checking shift closure conditions*

Set $\mathcal{C}^S = \mathcal{C}^S \cup \mathcal{C}_m^k$

set $\mathcal{C}^{NS} = \mathcal{C}^{NS} \setminus \mathcal{C}_m^k$

If $\mathcal{C}^{NS} = \emptyset$ **then go to Step 9**.

End If

If $\{\mathcal{C}_m^k = \emptyset \vee m \geq M-1\}$ **then**

If $\mathcal{U}^{NU} \neq \emptyset$ **then**

call POST_FIT_PROCEDURE

set $\mathcal{U}^{NU} = \mathcal{U}^{NU} \setminus \{u_k\}$

set $\mathcal{U}^{U} = \mathcal{U}^{U} \cup \{u_k\}$

and **go to Step 1**

else go to Step 9

End if

else set $m = m + 1$ and **go to Step 3**.

End if

Step 9. END

The POST_FIT_PROCEDURE looks for the smallest vehicle for the in-closure shift. Note that the procedure can be disabled whether the fleets are homogeneous.

POST_FIT_PROCEDURE:

select $u_h \in \mathcal{U}^{NU} s.t.\ Q^h = \min\{Q^l |\ Q^l \geqslant\ L_i^k)$
$\forall i = 1, \ldots, m\}$

If $\{\exists u_h\ \wedge Q^h < Q^k\}$ **then**

switch u_k with u_h

End if

At Step 0 the sets $\mathcal{U}^{NU}$, $\mathcal{U}^U$, $\mathcal{C}^S$, $\mathcal{C}^{NS}$ and k are initialized. If $k \leq K$, then at Step 1 the algorithm considers the first route $m = 1$ of the new $k-th$ shift, otherwise the algorithm stops.

At Step 2 the vehicle u_k with the biggest capacity is assigned to the shift.

Step 3 determines the seed of the cluster, i.e., the customer from which the composition of the cluster starts. The algorithm chooses the farthest customer position from the depot.

At Steps 4 and 5 the cluster is populated with the customer positions nearest to the seed. More precisely, Step 4 chooses the next node v_n and Step 5 checks the vehicle capacity constraint for the considered m-th route. If the capacity constraint is satisfied, then the new customer position v_n is included in the cluster and the procedure goes to Step 4 in order to determine a new position to be included; otherwise the cluster is provisionally considered as completed.

When the cluster is formed, the procedure solves a TSP instance by considering only the cluster nodes together with the starting and ending nodes according to equations (3.5.a) and (3.5.b). Indeed, Step 6 determines the optimal routing $\mathcal{R}_m^k$ as a solution of the farthest insertion heuristic.

Step 7 checks both the T_r and T_w constraints. If one of the two constraints is not satisfied, then the algorithm removes, from the cluster, the farthest node from the seed and goes to Step 6 in order to resolve again the TSP.

Step 8 updates sets $\mathcal{C}^S$ and $\mathcal{C}^{NS}$ and decides if there are the conditions to close or continue populating the current shift. More precisely, if the loop Step 6 - Step 7 emptied $\mathcal{C}_m^k$, it means that there were no time left for a new travel, and therefore the shift must be closed. Step 8 checks also the condition about the maximum number of travel per shift. In addition, for on-closing shifts, Step 8 checks if the vehicle u_k is the best choice for the shift. Indeed, if there is a different available vehicle u_h that can perform the shift with a lower capacity than the u_k capacity, then the algorithm associates u_h to the shift instead of u_k. We remark that it is not necessary to give the limit M: if M is very high, than the procedure returns, as an output, the number $\bar{m}$ of the routes performed by each vehicle. For the sake of clarity, Fig. 3.5 shows the Unified Modeling Language (UML) activity diagram [61] enlightening the steps of the proposed heuristic algorithm.

3.4 Postal Delivery Problem

3.4.1 Matematical Formulation

This section describes the configured VRP module of the U-DSS devoted to the PD for the city of Bari (Italy). The main activity of the PD system is the distribution of the postal products from the regional Post Automation Center (PAC) to the borough Postal Delivery Centers (PDCs).

The postal network is modeled by a fully connected directed graph G where the nodes are the PDCs and the depot is the PAC. The delivery service is performed by a single route for each shift ($M = 1$) in which every vehicle starts from and ends to the PAC (i.e., $v_N = v_0$).

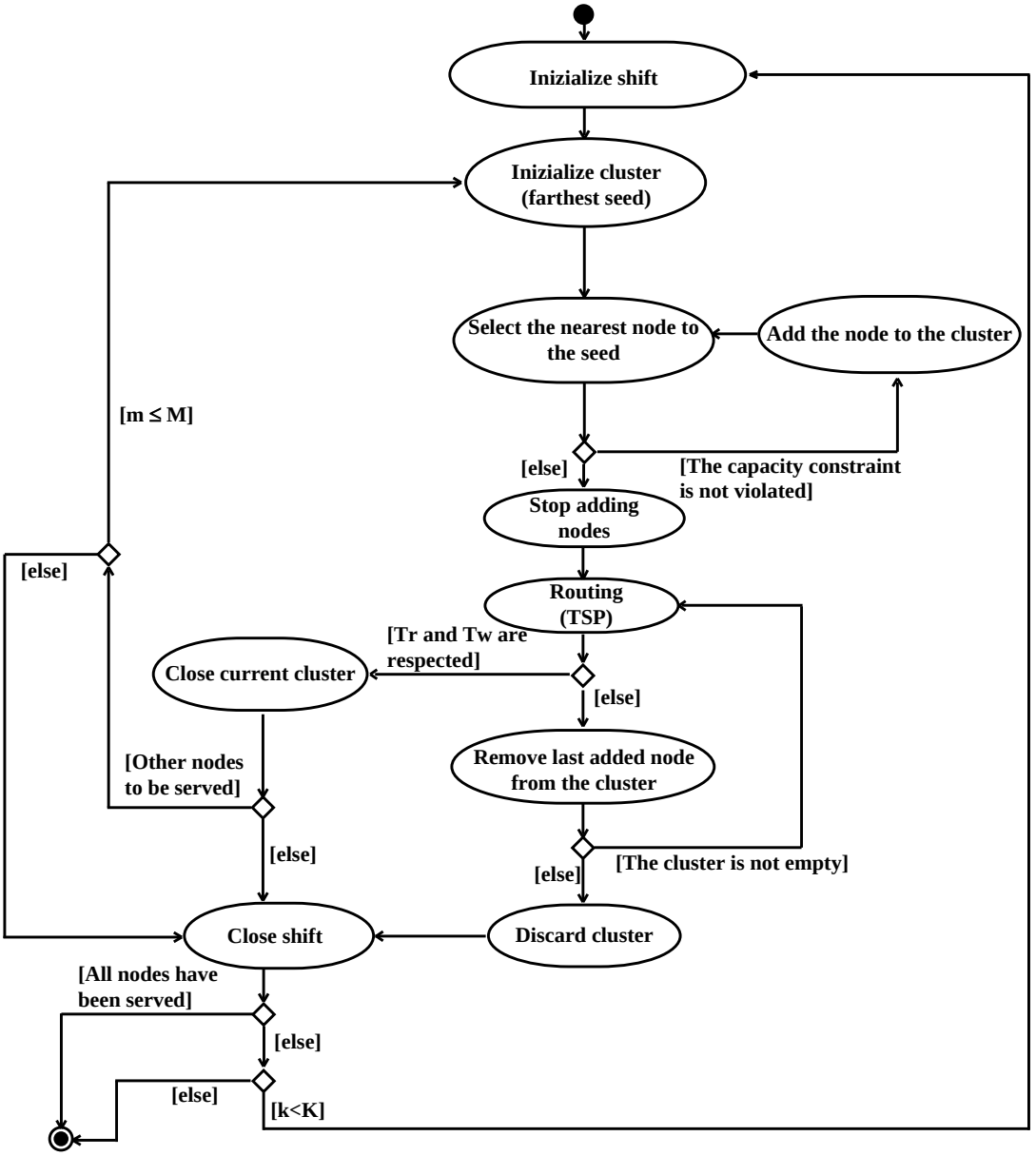

Figure 3.5: *UML Activity Diagram of the Vehicle Routing Algorithm.*

Now, the mathematical formulation to solve the VRP is presented in order to assess the proposed heuristic algorithm. The choice of the variables and the constraint structure is inspired by [62].

Decision variables:

$$z_{ij}^k = \begin{cases} 1, \text{ if vehicle } u_k \text{ visits in sequence nodes } v_i \text{ and } v_j \\ 0, \text{ otherwise.} \end{cases}$$

Dependent variables:

$L_{i,1}^k$ total load of vehicle u_k when it leaves node $v_i \in \mathcal{N}$ (it is (3.2) with $m = M = 1$);

S_i^k starting time of the mail delivery of vehicle u_k at node $v_i \in \overline{\mathcal{C}}$;

S_N^k working time of vehicle u_k.

The PD problem can be formulated as the following MILP problem.

$$\min \sum_{k=1}^{K} \sum_{i,j:e_{ij}\in\mathcal{E}} c_{ij} z_{ij}^k \tag{3.6}$$

$s.t.$

$$\sum_{k=1}^{K} \sum_{j:e_{ij}\in\mathcal{E}} z_{ij}^k = 1, \quad \forall v_i \in \overline{\mathcal{C}} \tag{3.6a}$$

$$\sum_{j:e_{0j}\in\mathcal{E}} z_{0j}^k = 1, \qquad \forall u_k \in \mathcal{U} \tag{3.6b}$$

$$\sum_{j:e_{iN}\in\mathcal{E}} z_{iN}^k = 1, \qquad \forall u_k \in \mathcal{U} \tag{3.6c}$$

$$\sum_{i:e_{i0}\in\mathcal{E}} z_{i0}^k = 0, \qquad \forall u_k \in \mathcal{U} \tag{3.6d}$$

$$\sum_{i:e_{ij}\in\mathcal{E}} z_{ij}^k - \sum_{j:e_{jh}\in\mathcal{E}} z_{jh}^k = 0, \qquad \forall u_k \in \mathcal{U}, \forall v_j \in \overline{\mathcal{C}} \tag{3.6e}$$

$$z_{ij}^k = 1 \Rightarrow\ L_{j,1}^k = L_{i,1}^k + q_j, \quad \forall u_k \in \mathcal{U},\ \forall e_{ij} \in \mathcal{E} \tag{3.6f}$$

$$z_{ij}^k = 1 \Rightarrow\ S_j^k \geq S_i^k + p_i + t_{ij}^k, \quad \forall u_k \in \mathcal{U},\ \forall e_{ij} \in \mathcal{E} \tag{3.6g}$$

$$0 \leq S_N^k \leq T_w, \quad \forall u_k \in \mathcal{U} \tag{3.6h}$$

$$0 \leq L_{i,1}^k \leq Q^k, \quad \forall u_k \in \mathcal{U} \quad \forall v_i \in \mathcal{N} \tag{3.6i}$$

$$z_{ij}^k \in \{0,1\} \tag{3.6j}$$

The objective function (3.6) minimizes the length of the routes performed by all the vehicles in each work shift. Constraints (3.6a) impose that each $v_i \in \overline{\mathcal{C}}$ is served exactly once. Constraints (3.6b)-(3.6c) guarantee that every vehicle starts the route from the depot and ends it at the node $v_N = v_0$. Constraints (3.6d) avoid that vehicles come back to v_0 after visiting $v_i \in \overline{\mathcal{C}}$ and (3.6e) ensure the flow conservation. Constraints (3.6f) and (3.6g) update the vehicle capacity and determine the arrival time of vehicle at each $v_i \in \mathcal{N}$, respectively Finally, (3.6h) and (3.6i) impose time and capacity constraints, respectively.

The outputs of the formulation are the routes assigned to the vehicles and the number of vehicles necessary to perform the PD.

3.4.2 Validation and Results

The performances of the MILP formulation and the heuristic algorithm are compared and tested by large sets of randomly generated instances. In this section we discuss six sets of them with five instances for each set. The characteristics of the data set are described in Table 3.1, where the values of N and K are chosen so that the MILP formulations can be solved in reasonable time.

The instances are generated by determining at random the values of q_i and c_{ij} according to a uniform distribution into the intervals given in Table 3.1. In addition, we consider a fleet composed by vehicles respectively of capacities 20, 15 and 10 Handling Units

(HUs), which is a capacity unit represented by the yellow box normally used for the mail delivery in Italy. In particular, the fourth, fifth and sixth column of Table 3.1 indicate the name of the vehicles associated to the corresponding capacities.

In order to determine the travel times t_{ij}^k, we assume an average speed of 50 km/h. Moreover, the service time p_i associated with $v_i \in \overline{\mathcal{C}}$ is 15 minutes and the vehicle working time is $T_w = 360$ minutes.

The MILP problem and the proposed heuristic are respectively implemented in Matlab environment and Java7, both on a PC equipped by a 3.40 Ghz Intel i7-3770 with 8 GB of memory, in single-thread mode. In particular, the MILP problem is solved by using the Gnu Linear Programming Kit [63].

Table 3.1: *Characteristics of Data Sets for Mail Service*

Sets	N [positions]	K [shifts]	20 [HU]	15 [HU]	10 [HU]	q_i [bins]	c_{ij} [km]
1	7	4	u_1	u_2,u_3	u_4	1-5	60-70
2	7	4	u_1	u_2,u_3	u_4	2-7	65-80
3	7	4	u_1	u_2,u_3	u_4	3-9	70-100
4	8	4	u_1	u_2,u_3	u_4	1-5	60-70
5	8	4	u_1	u_2,u_3	u_4	2-7	65-80
6	8	4	u_1	u_2,u_3	u_4	3-9	70-100

Table 3.2 summarizes the results that are obtained by allowing a maximum of 1000 seconds of CPU time for the solution of each instance: the objective function values of the MILP formulation (OPT), the objective function values of the heuristic algorithm (UB) and the CPU times to obtain them (CPU-OPT and CPU-UB, respectively).

Moreover, the last column of Table 3.2 reports the optimality gap (named *Gap*) expressed as follows:

$$Gap = (UB - OPT)/OPT * 100. \tag{3.7}$$

In particular, for the instances where the CPU time is equal to 1000 seconds, the optimal solution is not reached and the value of the best feasible obtained solution is reported.

Table 3.2: *Comparison between MILP Problem and Heuristic Algorithm*

Inst.	OPT	UB	CPU-OPT	CPU-UB	*Gap*
1.1	620.44 km	637.50 km	561 s	13 ms	2.75 %
1.2	621.61 km	636.01 km	332 s	11 ms	2.32 %
1.3	636.54 km	641.10 km	464 s	14 ms	0.72 %
1.4	632.14 km	632.14 km	527 s	12 ms	0.00 %
1.5	637.89 km	643.76 km	180 s	12 ms	0.92 %
2.1	775.75 km	778.48 km	340 s	14 ms	0.35 %
2.2	778.10 km	790.50 km	75 s	14 ms	1.59 %
2.3	762.09 km	772.69 km	480 s	15 ms	1.39 %
2.4	779.70 km	779.70 km	110 s	13 ms	0.00 %
2.5	775.59 km	776.94 km	68 s	14 ms	0.17 %
3.1	888.86 km	931.50 km	38 s	13 ms	4.80 %
3.2	861.79 km	873.17 km	23 s	11 ms	1.32 %
3.3	891.83 km	932.36 km	38 s	14 ms	4.54 %
3.4	854.98 km	862.17 km	62 s	13 ms	0.84%
3.5	853.14 km	869.85 km	66 s	14 ms	1.96%
4.1	695.26 km	708.22 km	1000 s	16 ms	1.86 %
4.2	696.62 km	699.42 km	1000 s	15 ms	0.40 %
4.3	686.71 km	693.95 km	1000 s	14 ms	1.05 %
4.4	687.75 km	707.89 km	796 s	13 ms	2.93 %
4.5	688.46 km	698.96 km	1000 s	14 ms	1.53 %
5.1	837.12 km	848.61 km	1000 s	15 ms	1.37 %
5.2	846.69 km	856.20 km	1000 s	16 ms	1.12 %
5.3	834.72 km	861.66 km	895 s	16 ms	3.23 %
5.4	828.48 km	828.48 km	923 s	13 ms	0.00 %
5.5	853.73 km	838.62 km	1000 s	12 ms	0.35 %
6.1	967.71 km	975.00 km	1000 s	13 ms	0.75 %
6.2	939.35 km	989.21 km	763 s	13 ms	5.31 %
6.3	956.05 km	981.45 km	1000 s	14 ms	2.66%
6.4	933.38 km	971.98 km	273 s	12 ms	4.14%
6.5	943.87 km	974.96 km	553 s	11 ms	3.29%

The results of Table 3.2 show that the value of the optimality gap, averaged on the 30 instances, is equal to 1.79 %, with a maximum value of 5.31 % and a variance of 2.27 %. Moreover, for 3 instances the two solutions coincide. In conclusion, the proposed heuristic algorithm obtains very good solutions in all considered instances and it runs in less than 17 ms.

3.4.3 Case Study of a Postal Delivery System

In this section, the U-DSS PD module is assessed by considering the case study of Bari. The network is modelled by the graph shown in Fig. 3.6 that depicts the locations of the nodes without showing the arcs for the sake of clarity. Hence, the network is composed by $N = 63$ nodes, where 61 nodes represent the PDCs and the node $v_0 = v_N$ is the PAC, which is symbolized by a red triangle in Fig. 3.6. So, the graph G is composed by $(N+1)^2 - (2N) = 3843$ directed arcs.

The values of the elements c_{ij} and t_{ij}^k are determined on the basis of the results obtained by Google Maps Application Programming Interface. The evaluation of these parameters is crucial for the routes optimization: indeed, they can be affected by the traffic congestion, the weather forecast, the number of street lanes, the slope of the streets, the considered period of the year and of the day.

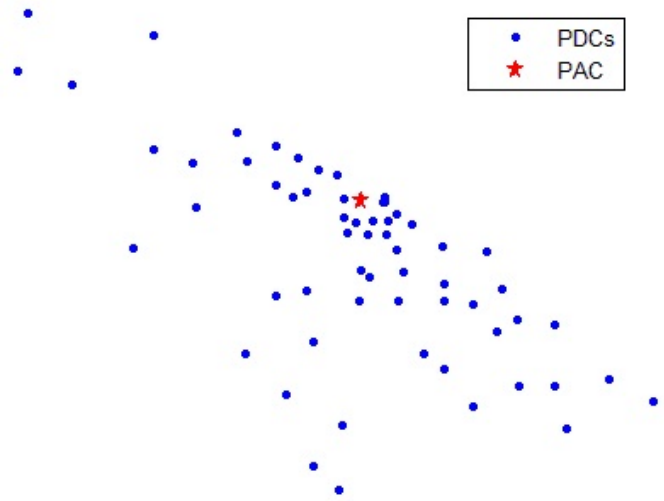

Figure 3.6: *The node locations of the postal delivery system of the case study.*

Moreover, the capacity constraints depend on the vehicle u_k used in the k-th shift. In

particular, the considered system has $K = 15$ vehicles with the capacities reported in Table 3.3: vehicles $u_1, ..., u_6$ have capacity equal to 20 HUs, $u_7, ..., u_{12}$ have capacity equal to 15 HUs and u_{13}, u_{14}, u_{15} have capacity equal to 10 HUs. The service duration in each node $v_i \in \overline{\mathcal{C}}$ is equal to $p_i = 15$ minutes, while the service duration in the PAC is $p_0 = p_N = 0$. The demand of each PDC daily changes: the values of the demand of each node that is considered in the case study is reported in Table 3.4. Since only one route can be performed in each shift, it holds $T_r = T_w = 360$ minutes.

Table 3.3: *Vehicle Fleet*

u_k	Q^k (HU)
$u_1 - u_6$	20
$u_7 - u_{12}$	15
$u_{13} - u_{15}$	10

The solution of the case study is summarized in Table 3.5 and Table 3.6. Let us remark that the first route is performed by the vehicle u_7 of capacity 15 HU because of the post fit procedure. Indeed, at first the heuristic algorithm tries to perform the route with the biggest available vehicle u_1. However, due to time constraints, the vehicle capacity is not used completely (14 HUs of 20), hence the algorithm switches u_1 with the vehicle u_7 of 15 HUs. In this way a better utilization of the vehicles is obtained.

Remark that in route 10 (see Table 3.5) the capacity of the vehicle is under utilized: this happens because there are not vehicles with capacity smaller than 10 HUs, and route 10 visits the last two residual nodes, near the depot. The decision maker could decide to assign those nodes to some of the other routes, by applying them an extra-time. The clusters are graphically represented in Fig. 3.7 with different colors: the red star represents the depot. For the sake of clarity, in Fig. 3.7 the connections with the depot are not depicted.

Table 3.4: *PDC Demand of the Case Study (in HU)*

v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
5	3	3	3	2	2	3	2	3	2	2
v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}
3	3	6	3	3	5	2	3	7	1	5
v_{23}	v_{24}	v_{25}	v_{26}	v_{27}	v_{28}	v_{29}	v_{30}	v_{31}	v_{32}	v_{33}
6	5	1	6	1	3	1	6	1	2	3
v_{34}	v_{35}	v_{36}	v_{37}	v_{38}	v_{39}	v_{40}	v_{41}	v_{42}	v_{43}	v_{44}
1	2	1	2	3	3	1	1	1	4	1
v_{45}	v_{46}	v_{47}	v_{48}	v_{49}	v_{50}	v_{51}	v_{52}	v_{53}	v_{54}	v_{55}
1	3	3	1	1	1	1	1	1	1	1
v_{56}	v_{57}	v_{58}	v_{59}	v_{60}	v_{61}					
1	5	1	1	2	1					

Table 3.5: *Route Characteristics of the Solution*

Route	Duration (min)	Length (km)	Load (HU)	Vehicle/Capacity
1	349.45	364.86	14	u_7/15 HU
2	296.92	313.84	18	u_1/20 HU
3	340.28	348.58	15	u_8/15 HU
4	341.53	322.43	19	u_2/20 HU
5	345.57	268.75	13	u_9/15 HU
6	335.52	234.89	18	u_3/20 HU
7	359.37	222.49	15	u_{10}/15 HU
8	317.72	154.65	18	u_4/20 HU
9	291.63	93.34	20	u_5/20 HU
10	66.15	28.10	2	u_{13}/10 HU
Sums	3044.13	2351.92	152	

Table 3.6: *Route Nodes of the Solution*

Route	$\mathcal{R}_1^k$
1	$v_0\ v_{38}\ v_{60}\ v_{61}\ v_{57}\ v_{39}\ v_0$
2	$v_0 v_{14} v_{26} v_{21} v_{22}\ v_0$
3	$v_0\ v_{33}\ v_{34}\ v_{35}\ v_{16}\ v_{30}\ v_0$
4	$v_0\ v_{23}\ v_{24}\ v_{17}\ v_{58}\ v_{32}\ v_0$
5	$v_0\ v_5\ v_{13}\ v_{27}\ v_{28}\ v_{29}\ v_3\ v_0$
6	$v_0\ v_{20}\ v_{25}\ v_{45}\ v_{43} v_{12}\ v_{44}\ v_{59}\ v_0$
7	$v_0\ v_{50}\ v_{11}\ v_{48}\ v_{31}\ v_{15} v_{49}\ v_{47}\ v_{46}\ v_0$
8	$v_0\ v_6\ v_{36}\ v_{56}\ v_{55}\ v_{19}\ v_4\ v_7\ v_{37}\ v_8\ v_0$
9	$v_0\ v_{53}\ v_{51}\ v_{42}\ v_{40}\ v_{41} v_{10}\ v_{18}\ v_2\ v_9\ v_1\ v_0$
10	$v_0\ v_{52}\ v_{54}\ v_0$

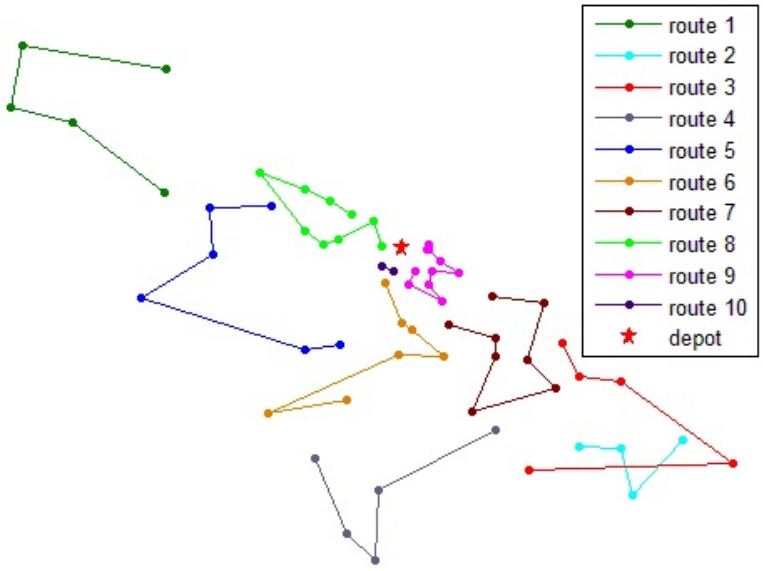

Figure 3.7: *The clusters obtained by the heuristic algorithm*

3.5 Waste Collection Problem

3.5.1 Waste Collection System Description and Matematical Formulation

In this section, the configured VRP module of the U-DSS devoted to a WC system is described. We consider a maximum number K of shifts and a set of K identical vehicles $\mathcal{U} = \{u_1, u_2, \dots u_K\}$: each vehicle $u_k \in \mathcal{U}$ can collect Q^k bins. Every day a team of workers drives the vehicles to perform the waste collection subject to time and capacity constraints. During a work shift, each driver of the team can perform several routes for a maximum of M, including the final return to the garage.

The first travel starts from the depot, follows the assigned route by collecting waste in bins and, when the truck is full or the time limit is attained, it has to stop at the landfill. In this site the workers unload the waste, make a stop and then start a new route from the landfill. On the basis of the company specifications, the work shift goes to an end by an empty travel from the landfill to the garage. Each route time has to be less than or equal to T_r minutes and the shift is limited by the working time of T_w minutes.

Fig. 3.8 shows the UML activity diagram of the vehicle shift highlighting the actions of drivers and vehicles.

The problem is minimizing the length of all the routes performed by the vehicles in each work shift. Each route is given as a sequence of visited nodes with the indication of the arrival times.

Now, a mathematical formulation of the considered VRP is presented on the basis of the model in [62] but also providing several routes in each shift. Hence, the decision and dependent variables are slightly modified with respect to the MILP (3.6).

Decision variables:

$$z^k_{ij,m} = \begin{cases} 1, \text{ if vehicle } u_k \text{ visits in sequence nodes } v_i \text{ and } v_j \\ \quad \text{during route } m \text{ of shift } k \\ 0, \text{ otherwise.} \end{cases}$$

Dependent variables:

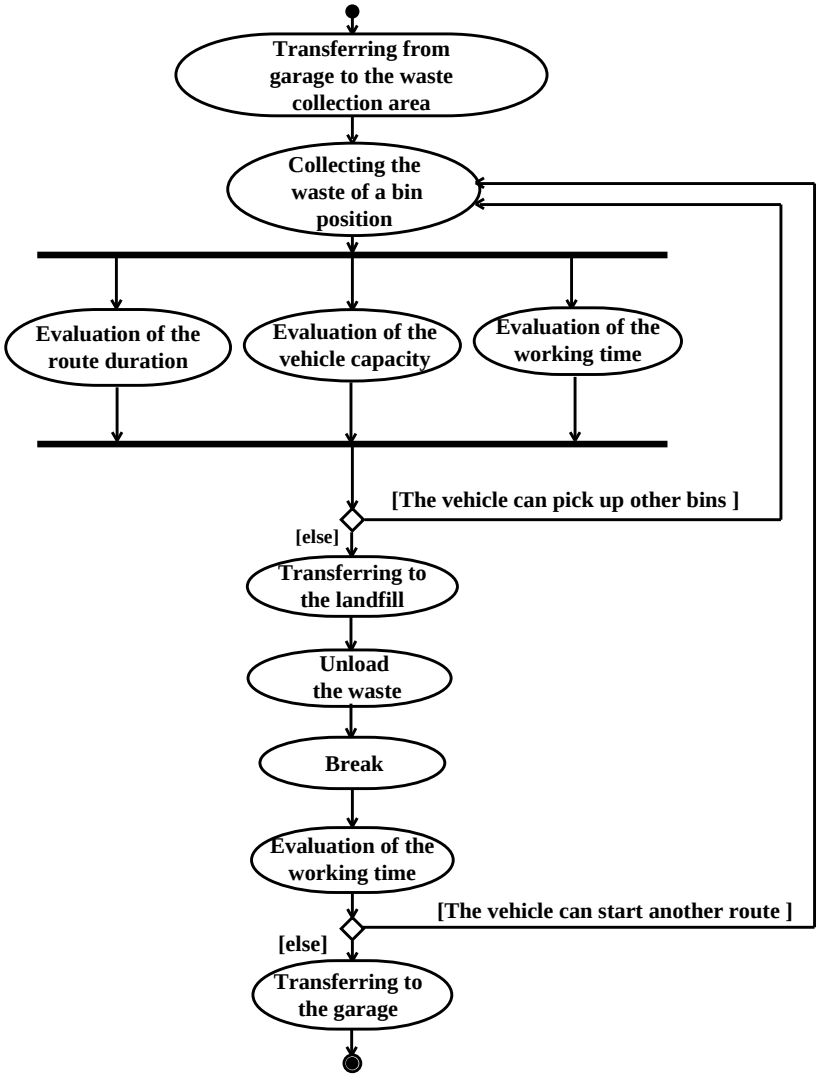

Figure 3.8: *Activity diagram of the vehicle during a shift.*

$L^k_{i,m}$ load collected by vehicle u_k when it leaves node $v_i \in \mathcal{N}$ during route m (see (3.2));

$S^k_{i,m}$ starting time of the pick up operation of vehicle u_k at node $v_i \in \overline{\mathcal{C}}$ during route m;

$S^k_{N,m}$ arrival time to node v_N of route m performed by u_k;

$S^k_{0,m}$ working time of vehicle u_k during route m.

The aim of the considered VRP is to design a set of feasible routes with minimum total length such that:

- each node is visited exactly once by one vehicle;
- the first route starts from v_0 and ends to v_N;
- the last (empty) route starts from v_N and ends to v_0;
- the remaining routes start from and end to v_N;
- the number of routes performed by each vehicles is less than or equal to M.

Now, the considered VRP can be formulated by the following MILP problem.

$$\min \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{i,j:e_{ij}\in\mathcal{E}} c_{ij} z^k_{ij,m} \tag{3.8}$$

$s.t.$

$$\sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{j:e_{ij}\in\mathcal{E}} z^k_{ij,m} = 1, \qquad \forall v_i \in \overline{\mathcal{C}} \tag{3.8a}$$

$$\sum_{j:e_{0j}\in\mathcal{E}} z^k_{0j,1} = 1, \qquad \forall u_k \in \mathcal{U} \tag{3.8b}$$

$$\sum_{m=2}^{M} \sum_{j:e_{0j}\in\mathcal{E}} z^k_{0j,m} = 0, \qquad \forall u_k \in \mathcal{U} \tag{3.8c}$$

$$\sum_{j:e_{Nj}\in\mathcal{E}} z^k_{Nj,m} \leq 1, \quad \forall u_k \in \mathcal{U},\ m = 2,\dots,M \tag{3.8d}$$

$$\sum_{i:e_{ij}\in\mathcal{E}} z^k_{ij,m} - \sum_{i:e_{jh}\in\mathcal{E}-\{e_{j0}\}} z^k_{jh,m} = 0, \tag{3.8e}$$

$$\forall v_j \in \overline{\mathcal{C}},\ \forall u_k \in \mathcal{U},\ \forall m$$

$$\sum_{i:e_{iN}\in\mathcal{E}} z^k_{iN,m} - \sum_{j:e_{Nj}\in\mathcal{E}} z^k_{Nj,m+1} = 0, \tag{3.8f}$$

$$\forall u_k \in \mathcal{U}, m = 1,2,\dots,M-1$$

$$\sum_{m=1}^{M} z^k_{N0,m} = 1, \qquad \forall u_k \in \mathcal{U} \tag{3.8g}$$

$$z^k_{ij,m} = 1 \Rightarrow\ S^k_{j,m} \geq S^k_{i,m} + p_i + t^k_{ij}, \tag{3.8h}$$

$$\forall u_k \in \mathcal{U}, \forall e_{ij} \in \mathcal{E}\ s.t.\ i \neq N,\ \forall m$$

$$z^k_{Nj,m} = 1 \Rightarrow\ S^k_{j,m} \geq S^k_{N,m-1} + p_N + t^k_{Nj}, \tag{3.8i}$$

$$\forall u_k \in \mathcal{U},\ \forall j\ s.t.\ e_{Nj} \in \mathcal{E},\ \forall m > 1$$

$$z^k_{ij,m} = 1 \Rightarrow\ L^k_{j,m} = L^k_{i,m} + q_j, \tag{3.8j}$$

$$\forall u_k \in \mathcal{U},\ \forall e_{ij} \in \mathcal{E}\ s.t.\ i \neq N,\ \forall m$$

$$z^k_{Nj,m} = 1 \Rightarrow\ L^k_{j,m} = q_j, \tag{3.8k}$$

$$\forall u_k \in \mathcal{U},\ \forall j\ s.t.\ e_{Nj} \in \mathcal{E},\ \forall m > 1$$

$$0 \leq S^k_{N,1} \leq T_r, \qquad \forall u_k \in \mathcal{U} \tag{3.8l}$$

$$0 \leq S^k_{N,m} - S^k_{N,m-1} - p_N \leq T_r, \quad \forall u_k \in \mathcal{U}, \forall m > 1 \tag{3.8m}$$

$$0 \leq S^k_{0,m} \leq T_w, \qquad \forall u_k \in \mathcal{U},\ \forall m \tag{3.8n}$$

$$0 \leq L^k_{i,m} \leq Q^k, \quad \forall u_k \in \mathcal{U},\ \forall v_i \in \mathcal{N},\ \forall m \tag{3.8o}$$

$$z^k_{ij,m} \in \{0,1\} \tag{3.8p}$$

The objective function (3.8) minimizes the length of the routes performed by all the vehicles in each work shift.

Constraints (3.8a) impose that each node $v_i \in \overline{\mathcal{C}}$ has to be served exactly once. The constraints (3.8b)-(3.8d) guarantee that every vehicle starts the first route at the garage

while the other routes begin from the disposal site. Constraints (3.8e) ensure the flow conservation avoiding that the vehicles come back to the garage after having served a node $v_i \in \overline{\mathcal{C}}$. Constraints (3.8f) connect the end of route m to the start of route $m+1$, while assuring that all routes but the last end at the landfill site. Constraints (3.8g) impose that the last route of a shift has to be performed from the disposal site to the garage.

Constraints (3.8h) and (3.8i) determine the arrival time at node $v_i \in \mathcal{N}$ and constraints (3.8j) and (3.8k) update the loads $L^k_{i,m}$. On the other hand, (3.8l)-(3.8o) impose the time and capacity constraints.

Note that the non-linear constraints (3.8h)-(3.8k) can be linearised by a big number formulation [64].

3.5.2 Validation and Results

The performances of the MILP formulation and the heuristic algorithm are compared and tested by nine sets of randomly generated instances. Table 3.7 reports the data sets chosen to compare the two approaches: the values of N and K are selected so that the MILP formulations can be solved in reasonable time. Note that the instances of the sets 1-3, 4-6 and 7-9 generate 720, 891 and 1440 variables, respectively. Moreover, the sets 1-3 as well as the sets 4-6 and 7-9 differ for the demand in each pick up position and for the distance between the nodes: these parameters are generated with uniform probabilities between the values shown respectively in the forth and fifth columns of Table 3.7.

In order to determine the route times t^k_{ij} between two consecutive pick up sites, we assume a speed of 50 km/h in the following cases: i) the vehicle goes from the depot to the first not served pick up point; ii) the vehicle goes from the landfill to the first not served pick up point; iii) the vehicle comes back to the landfill from the last served pick up point; iv) the vehicle comes back to the garage from the landfill. In all the other cases the average speed is assumed equal to 25 km/h.

Moreover, the service time p_i associated with $v_i \in \overline{\mathcal{C}}$ is determined on the basis of the number of bins located in the considered pick up position: the vehicle simultaneously loads two bins in 1.5 minutes. On the other hand, the unloading time at the landfill is

$p_N = 30$ minutes and includes the compulsory work break. Moreover it holds $q_0 = q_N = 0$ and $p_0 = 0$.

Summing up, the route times t_{ij}^k for $e_{ij} \in \mathcal{E}$ and the service times p_i for $v_i \in \mathcal{N}$ are determined by the following relations:

$$t_{ij}^k = \begin{cases} \frac{c_{ij} \cdot 60}{25000}, \ \forall i, j \text{ with } i \neq 0, N \text{ or } j \neq 0, N \\ \\ \frac{c_{ij} \cdot 60}{50000}, \text{otherwise} \end{cases} \tag{3.9}$$

$$p_i = \begin{cases} 0, \text{ if } i = 0; \\ 30, \text{ if } i = N; \\ 1.5\lceil \frac{q_i}{2} \rceil, \text{otherwise.} \end{cases} \tag{3.10}$$

Table 3.7: *Characteristics of Data Sets*

Sets	N [positions]	K [vehicles]	q_i [bins]	c_{ij} [kilometers]
1	6	3	30-40	20-30
2	6	3	35-45	15-25
3	6	3	30-50	10-40
4	7	3	30-40	20-30
5	7	3	35-45	15-25
6	7	3	30-50	10-40
7	8	4	30-40	20-30
8	8	4	35-45	15-25
9	8	4	30-50	10-40

Table 3.8 shows the results that are obtained by allowing a maximum of 1000 seconds of CPU time for the solution of each instance.

Table 3.8: *Comparison between MILP Problem and Heuristic Algorithm*

Inst.	OPT	UB	CPU-OPT	CPU-UB	*Gap*
1.1	324.67 km	337.36 km	24 s	26 ms	3.91 %
1.2	326.62 km	336.44 km	29 s	23 ms	3.01 %
1.3	327.77 km	335.43 km	25 s	28 ms	2.34 %
1.4	296.78 km	302.26 km	12 s	23 ms	1.85 %
1.5	336.92 km	336.92 km	25 s	23 ms	0.00 %
2.1	270.55 km	276.12 km	21 s	25 ms	2.06 %
2.2	274.48 km	277.08 km	18 s	25 ms	0.95 %
2.3	250.38 km	260.09 km	25 s	23 ms	3.88 %
2.4	273.56 km	291.48 km	22 s	25 ms	6.55 %
2.5	260.55 km	281.83 km	18 s	26 ms	8.16 %
3.1	262.48 km	270.42 km	6 s	23 ms	3.02 %
3.2	294.96 km	319.06 km	14 s	26 ms	8.17 %
3.3	228.17 km	228.17 km	6 s	20 ms	0.00 %
3.4	258.15 km	261.31 km	26 s	23 ms	1.25 %
3.5	267.62 km	281.27 km	13 s	23 ms	5.10 %
4.1	376.36 km	386.56 km	102 s	23 ms	2.71 %
4.2	369.89 km	387.03 km	265 s	29 ms	4.63 %
4.3	379.81 km	386.00 km	232 s	26 ms	1.89 %
4.4	364.07 km	375.68 km	84 s	24 ms	3.19 %
4.5	374.37 km	433.83 km	166 s	28 ms	15.77 %
5.1	320.66 km	345.12 km	98 s	31 ms	7.63 %
5.2	314.91 km	342.94 km	120 s	29 ms	8.91 %
5.3	309.72 km	348.84 km	29 s	26 ms	12.63 %
5.4	312.38 km	352.76 km	41 s	27 ms	12.93 %
5.5	323.08 km	363.32 km	137 s	25 ms	12.45 %
6.1	369.20 km	435.14 km	11 s	26 ms	17.86 %

(Continued on next page)

Table 3.8 – *(continued from previous page)*

Inst.	OPT	UB	CPU-OPT	CPU-UB	*Gap*
6.2	353.40 km	369.41 km	30 s	26 ms	4.53 %
6.3	338.60 km	393.52 km	32 s	27 ms	16.22%
6.4	305.49 km	336.14 km	8 s	27 ms	10.03%
6.5	314.21 km	349.04 km	13 s	25 ms	11.08%
7.1	417.08 km	434.45 km	1000 s	26 ms	4.16 %
7.2	414.14 km	422.70 km	1000 s	26 ms	1.95 %
7.3	421.70 km	431.66 km	1000 s	27 ms	2.36 %
7.4	441.12 km	462.74 km	843 s	29 ms	4.90 %
7.5	442.35 km	476.02 km	748 s	31 ms	7.61 %
8.1	349.10 km	366.66 km	1000 s	28 ms	5.03 %
8.2	344.15 km	357.50 km	1000 s	29 ms	3.88 %
8.3	367.87 km	377.50 km	1000 s	28 ms	2.62 %
8.4	363.33 km	378.44 km	1000 s	26 ms	4.16 %
8.5	348.70 km	357.70 km	1000 s	28 ms	2.58 %
9.1	351.98 km	378.11 km	1000 s	28 ms	7.42 %
9.2	373.88 km	404.99 km	1000 s	27 ms	8.55 %
9.3	383.16 km	418.24 km	725 s	31 ms	9.15%
9.4	403.37 km	428.82 km	1000 s	31 ms	6.31%
9.5	413.37 km	428.56 km	1000 s	28 ms	3.67%

In particular, we remember that for the instances where the CPU time is equal to 1000 seconds the optimal solution has been not reached and the value of the best feasible obtained solution is reported. The results of Table 3.8 show that the value of the optimality gap (3.7), averaged on the 45 instances, is equal to 5.93 %, with a maximum value of 17.86 %. Moreover, it is important to note that *Gap* is less than 10 % for 37 instances, it is less than 5% for 25 instances and that for 2 instances the two solutions coincide. Considering only the 33 optimal solutions, the *Gap* is less than 10 % for 25 instances.

The *Gap* of the instances 4.5 and 6.1 is justified by the fact that the MILP problem solution provides three vehicles, while the heuristic algorithm provides four vehicles: this difference determines the largest gap between the two solutions.
In conclusion, the proposed heuristic algorithm obtains good solutions in the most of the considered instances and runs in less than 31 ms.

3.5.3 Case Study of a Waste Collection System

In this subsection the case study of WC problem of city of Trieste is described and solved, with respect to data of early 2015. The maximum number of shifts is $K = 20$ that is the number of the shifts performed in the current waste management. Moreover, 20 identical vehicles form the available fleet with capacity $Q^k = 102$ bins for $k = 1, ..., 20$. During a work shift, each driver of the team can perform a maximum number of $M = 3$ routes and it holds $T_r = 200$ and $T_w = 360$ minutes.
The waste collection system of the considered city consists of 2,790 bins distributed over $N-2 = 1,498$ pick up positions. The graph G is of 1,493 nodes and $(N+1)^2-(N+1) = 2,227,556$ arcs. In order to perform a realistic evaluation of the routes, each value of c_{ij} is increased of 0.15 km and 0.30 km for each left-turn and U-turn, respectively.
The solution of the considered WC obtained by the heuristic algorithm is shown in Table 3.9. It is apparent that 17 shifts and 35 routes are necessary to perform the waste collection. More precisely, most of the shifts are composed of two routes, not considering the fixed empty route from the landfill to the garage at the end of the shift. Although these "end-shift" routes are omitted in Table 3.9, total duration and total length take them in account (each of them has a duration of about 7 minutes for a length of 4.5 kilometers).

Table 3.9: *Solution Obtained by the Heuristic Algorithm*

u_k	m	duration [minutes]	length [kilometers]	Loaded bins [bins]
u_1	1	199.7	62.6	69

(Continued on next page)

Table 3.9 – *(continued from previous page)*

u_k	m	duration [minutes]	length [kilometers]	Loaded bins [bins]
	2	92.6	37.8	32
u_2	1	198.0	50.7	89
	2	95.6	33.0	34
u_3	1	199.8	44.4	98
	2	90.8	32.2	36
u_4	1	198.9	47.1	89
	2	95.1	34.7	27
u_5	1	162.8	32.8	102
	2	126.5	34.3	57
u_6	1	162.3	31.5	101
	2	130.9	30.8	80
u_7	1	153.1	28.7	101
	2	140.1	28.9	92
u_8	1	162.5	24.6	102
	2	131.1	25.7	88
u_9	1	191.3	36.9	102
	2	99.3	22.0	69
u_{10}	1	158.7	24.8	101
	2	126.7	25.9	102
u_{11}	1	135.5	23.9	101
	2	149.0	29.2	102
u_{12}	1	181.0	29.3	99
	2	110.6	24.7	79
u_{13}	1	135.3	22.6	102
	2	132.5	21.6	101

(Continued on next page)

Table 3.9 – *(continued from previous page)*

u_k	m	duration [minutes]	length [kilometers]	Loaded bins [bins]
u_{14}	1	127.1	21.9	99
	2	147.8	23.0	99
u_{15}	1	153.3	28.1	102
	2	127.8	18.6	102
u_{16}	1	132.0	20.4	102
	2	161.8	26.9	97
u_{17}	1	28.7	7.8	17
Total		5713.5	1058.5	2773

In order to show the effectiveness of the proposed algorithm, the obtained solution is compared with the current waste management of the case study, where 20 shifts composed of 2 routes (not considering the last route) are currently performed. Hence, the proposed solution allows saving 3 shifts with significant economic and environmental benefits.

Finally, Fig. 3.9 and Fig. 3.10 show all clusters and a shift respectively. In particular, clusters belonging to the same shift are denoted with different tones of the same color.

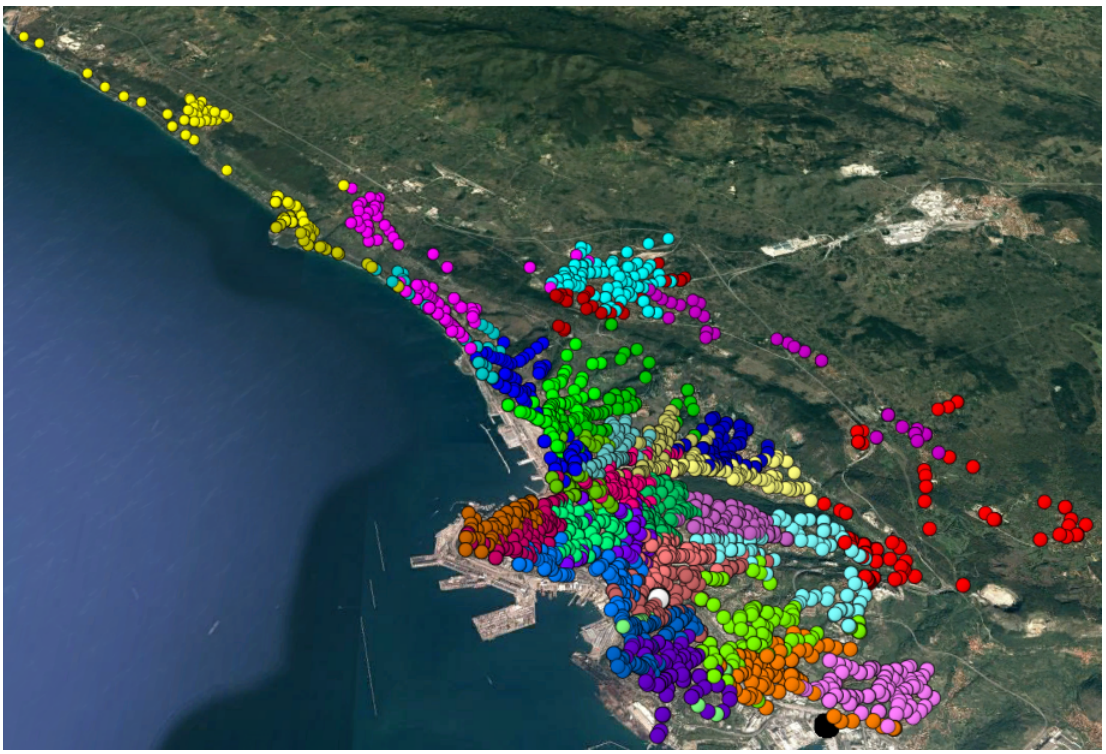

Figure 3.9: *Clusters in shifts*

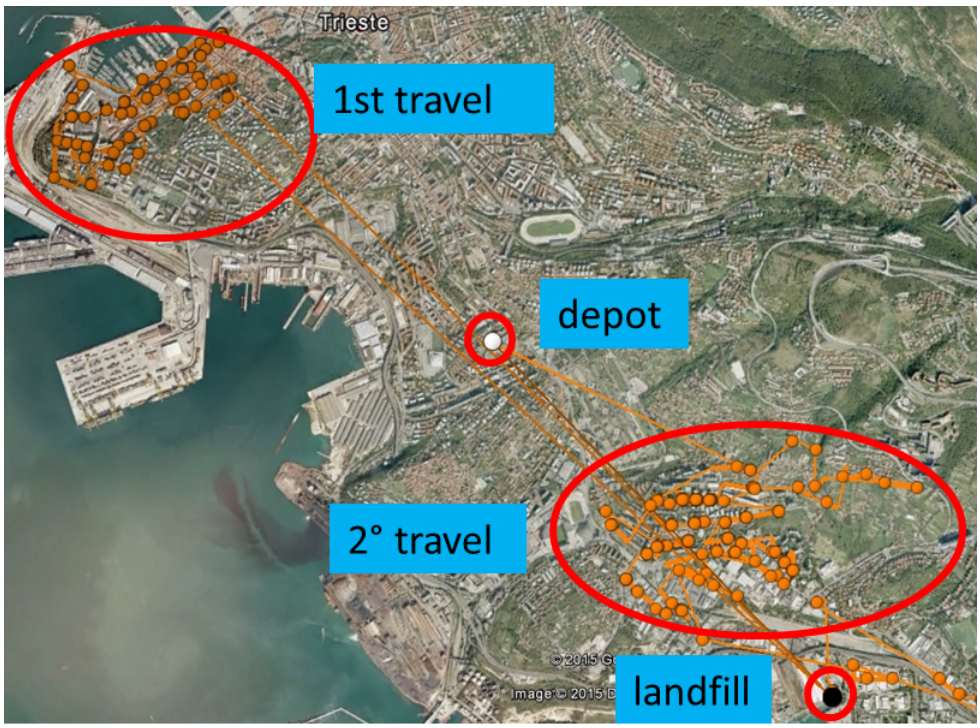

Figure 3.10: *A particular shift*

Chapter 4

A Distributed Approach for Vehicle Routing Problem with Time Windows

The aim of this chapter is to propose a distributed solution strategy for vehicle routing problems with time windows that could provide twofold key features. First, an iterative and distributed algorithm enables vehicles to compute routes that are characterized by balanced workloads for the drivers. Second, the method is applicable to plan the daily workload of vehicles and to adapt the routing plan for servicing ongoing requests.

4.1 Introduction

VRPTW deals with customers accepting to be served by vehicles with limited capacity only in prefixed time intervals, called time windows. Time windows can be: i) *soft*, when vehicles can serve the customers outside of the time windows with a penalty affecting the objective function value; ii) *hard*, when vehicles are forbidden to serve a customer out of its time window.

The problem of the time windows constraints is studied in [65], a review paper that points out that while computing the possible inter-node trips can be managed efficiently, checking the satisfaction of the time windows constraints can be the bottleneck of the

algorithm.

The VRP and the VRPTW are NP-hard problems [66], then it is necessary to propose alternative resolution methods, in particular to solve the problem in very large networks. The *cluster first, route second* method is a well established solution for solving the VRP [57, 67, 68] and the VRPTW: [69] proposes a cluster first, route second hierarchical hybrid approach which also takes in account the time windows constraint. However, in such approaches, which are centralized as the most part of literature, the vehicle routes are not balanced. More precisely, it happens that some vehicles have to perform long routes and other short routes, with a distribution of workloads among drivers that are unrealisable in practice.

This chapter presents a new approach for the deterministic static and dynamic VRPTW problems solution with a *cluster first, route second* method. The clustering step is formalized as a distributed graph partitioning problem, extended with time windows constraints and modeled by an ILP problem, which allows vehicles assigning themselves autonomously unique clusters of customers. The approach exploits contributions about consensus and distributed task assignment by [7] and [8]. Finally, due to the division between clustering and routing phases, the latter is performed in a decentralized approach by each vehicle, which solves an instance of TSPTW considering only the customers belonging to the associated cluster.

Furthermore, such clustering approach allows obtaining particular features of the customer clusters: the subsequent routing phase, applied to the clusters, provides balanced routes with respect to the vehicle traveling times and assigned loads.

The methodology can be applied in two phases of the pick up services: i) in the first phase, at the beginning of the working day, when the empty vehicles have to plan their routes starting from the central depot (static problem); ii) in the second phase to manage the requests arrived during the travel (dynamic problem). Indeed, the chapter considers a VRPTW problem and a MDVRPTW that can act as components of a *deterministic* Dynamic VRPTW: the former represents the problem to be solved in order to compute the initial routing plan, when all the vehicle are at a central depot; the latter represents the situation at a certain point of the execution of the plan, when vehicles have already

left the central depot, but new dynamic customers requests arrive.
The chapter is organized as follows. Section 4.2 briefly introduces the problem and the general strategy. Sections 4.3, 4.4 and 4.5 present problem models and the approaches proposed for solving the VRPTW and MDVRPTW, which act as static and dynamic phase respectively. Finally, Section 4.6 discusses a set of benchmark problem solutions while Section 4.7 enlightens the applicability to an example inspired to a courier service company.

4.2 Problem Characterization and Strategy

We consider a set $\mathcal{U}$ of U homogeneous vehicles of capacity Q. The vehicles are in a central depot before the beginning of the working day, which can be considered as starting at time $\tau = 0$. Before this time, the C customers belonging to the set $\mathcal{C}$ have placed their requests and a static routing plan is prepared: each customer must be served by only one vehicle, which can visit the customer only once in its route. The working day ends at time H.
However, during the working day $0 < \tau \leq H$, some other requests can be placed by new customers and the vehicles need to adapt their routes in order to take into account these new nodes to be served, as well as the already served requests and the residual capacities. In other words, the set $\mathcal{C}$ and the plan of the fleet evolve during this dynamic phase.
In the static and dynamic phases of the VRP, each customer $i \in \mathcal{C}$ is described by the following parameters:

- the couple $coords_i = (x_i, y_i)$ denotes the position of customer i in a common reference system;
- the demand amount q_i represents the quantity of goods that customer i has to load on the assigned vehicle;
- the time window $[a_i, b_i]$ defines the early time a_i and the late time b_i, i.e., the vehicle assigned to i has to start service at the customer between a_i and b_i;

- the service time p_i denotes the time required by the assigned vehicle to satisfy the request of customer i, after service has started.

There is only one difference among the customers whose requests are known before $\tau = 0$ and the other ones: the former has hard time windows, while the latter has soft time windows, i.e., a vehicle can serve a customer i before a_i with a penalty cost that is proportional to the time advance by a coefficient γ.

Fig. 4.1 presents a scheme of the proposed heuristic decentralized approach to the VRPTW problem based on the *cluster first, route second* method. The approach solves the pick up services considering two problems: static and dynamic, both divided in two phases.

First, the static problem is solved before the working day starts: the vehicles, which are considered as autonomous intelligent agents, solve in the clustering phase a distributed graph partitioning problem by means of local ILP problems; then, in the TSPTW phase, a set of customers is associated with each vehicle. At this point, each vehicle can determine, in a decentralized approach, the route to be followed in order to serve each customer of its cluster in the constrained time windows.

Second, the dynamic problem is solved during the working hours $\tau^* < \tau \leq H$: at a certain time τ^*, the vehicles solve a distributed graph partitioning problem with multiple depots. In fact, for the new optimization, each vehicle $k \in \mathcal{U}$ is in a different position that can be considered as a virtual departure depot $d_i \in \mathcal{D}$ with $|\mathcal{D}| = |\mathcal{U}|$.

4.3 The Static Vehicle Routing Problem with Time Windows

This section focuses on the VRPTW problem devoted to the initial routing plan. First, a VRPTW formulation is shown. Secondly, a centralized graph partitioning problem with time windows is introduced as a model for the subsequent distributed approach, based on the *cluster first, route second* method. Thirdly, a distributed algorithm for the clustering phase is illustrated. Finally, the routing phase is introduced.

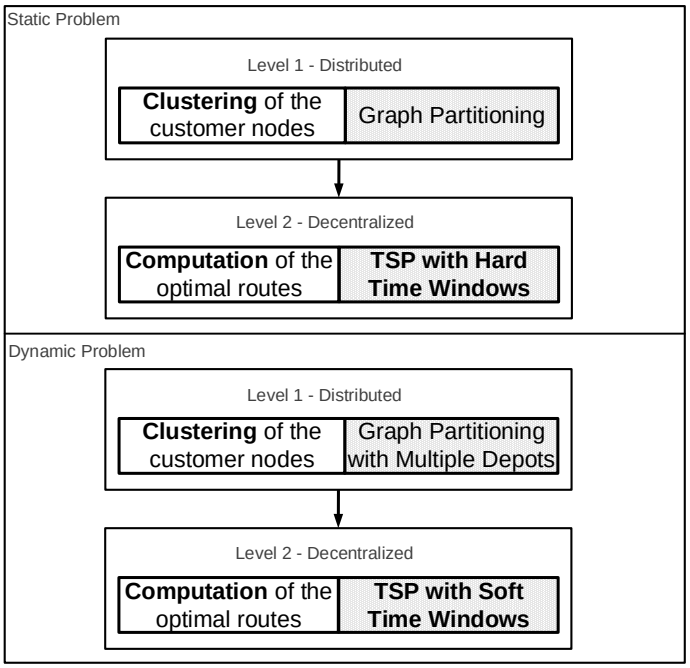

Figure 4.1: *Architecture of the overall proposed approach*

4.3.1 Problem Formulation

In this section we present a MIP formulation of the static VRPTW [70].

The network of customers is modeled by a fully connected indirect graph $G = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes composed by the set of customers, the initial depot 0 and the final depot $(C+1)$ which can be equal to the initial one, i.e., $\mathcal{N} = \mathcal{C} \cup \{0, (C+1)\}$. Moreover, the set of graph edges is $\mathcal{E} = \{(i,j) |\, i,j \in \mathcal{N},\ i \neq j\}$, where edge $(i,j) \in \mathcal{E}$ if there is a direct connection between i and j. In addition, a cost t_{ij} is associated to each edge $(i,j) \in \mathcal{E}$, representing the time required by a vehicle to cover the distance between i and j. Moreover, the time $\theta_{ij} = t_{ij} + p_i$ includes the cost and the service time of customer i.

Each vehicle $k \in \mathcal{U}$, with capacity Q, starts its trip from node $i = 0$, ends it in node $i = C+1$ and visits a sequence of customers $i \in \mathcal{C}$ by satisfying their demand amount q_i and the time window $[a_i, b_i]$. The plan has to not exceed the planning horizon $[0, H]$, i.e., the following assumption is considered: $[a_0, b_0] = [a_{C+1}, b_{C+1}] = [0, H]$.

The decision variables are defined as follows:

$$z_{ij}^k = \begin{cases} 1, & \text{if } k \text{ visits } j \in \mathcal{N} \text{ directly after } i \in \mathcal{N} \\ 0 & \text{otherwise,} \end{cases} \tag{4.1}$$

$$s_{ik} \in \mathbb{R}^+ \text{ start service time of } k \in \mathcal{U} \text{ at node } i \in \mathcal{N} \tag{4.2}$$

The problem aims at minimizing the total travel time of the vehicle fleet.
Now, the following MIP problem models the described VRPTW:

$$\begin{aligned} &\min \sum_{k\in\mathcal{U}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} t_{ij} z_{ij}^k \\ &s.t. \end{aligned} \tag{4.3}$$

$$\sum_{k\in\mathcal{U}} \sum_{j\in\mathcal{N}} z_{ij}^k = 1, \ \forall\, i \in \mathcal{C} \tag{4.3a}$$

$$\sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{N}} q_i z_{ij}^k \leq Q, \ \forall\, k \in \mathcal{U} \tag{4.3b}$$

$$\sum_{j\in\mathcal{N}} z_{0j}^k = 1, \ \forall\, k \in \mathcal{U} \tag{4.3c}$$

$$\sum_{i\in\mathcal{N}} z_{ih}^k - \sum_{j\in\mathcal{N}} z_{hj}^k = 0, \ \forall\, h \in \mathcal{C}, \ \forall\, k \in \mathcal{U} \tag{4.3d}$$

$$\sum_{i\in\mathcal{N}} z_{i,C+1}^k = 1, \ \forall\, k \in \mathcal{U} \tag{4.3e}$$

$$z_{ij}^k (s_{ik} + \theta_{ij} - s_{jk}) \leq 0, \ \forall\, i, j \in \mathcal{N}, \ \forall\, k \in \mathcal{U} \tag{4.3f}$$

$$a_i \leq s_{ik} \leq b_i, \ \forall\, i \in \mathcal{N}, \ \forall\, k \in \mathcal{U} \tag{4.3g}$$

$$z_{ij}^k \in \{0, 1\}, \ \forall\, i, j \in \mathcal{N}, \ \forall\, k \in \mathcal{U} \tag{4.3h}$$

The objective function of the MIP problem (4.3) minimizes the total travel time of the vehicle fleet. Constraints (4.3a) state that each customer can be visited exactly once.

Constraints (4.3b) do not allow vehicles to be loaded with more than their capacities. Constraints (4.3c), (4.3d), (4.3e) are flow equations also ensuring that a vehicle starts its trip from node $i = 0$ and ends it at $i = C + 1$. A constraint of type (4.3f) forces a vehicle, which has to go from customer i to customer j, to start service at j only after serving i and traveling between the two. Lastly, constraints (4.3g) impose time windows restrictions for each customer. Note that for each vehicle the service start variables s_{ik} impose a unique route direction thereby eliminating any subtours [70].
Finally, note that, strictly, (4.3) is not a MIP problem because of the non linear restrictions (4.3f), whose expression makes time window constraints more readable. However, (4.3f) can be linearized by the big-M technique as follows [70]:

$$s_{ik} + \theta_{ij} - M(1 - z_{ij}^k) \leq s_{jk},\ \forall\, i, j \in \mathcal{N},\ \forall\, k \in \mathcal{U},$$

where M is a large scalar whose value has to satisfy the following:

$$M \geq \max_{i,j \in \mathcal{N}} \{b_i + \theta_{ij} - a_j\}$$

4.3.2 The Clustering Phase: a Centralized Formulation

As aforementioned, the overall aim of this section is to propose a heuristic approach to the static VRPTW problem based on the *cluster first, route second* method. First, the vehicles solve a graph partitioning problem [71] extended with time windows and modeled by local ILP problems. After the clustering phase, a set of customers is associated with each vehicle. Then, each vehicle determines, in a decentralized way, the route to be followed in order to serve each customer of its cluster in the constrained time windows (the TSPTW phase).
However, the starting point of the distributed approach is a centralized formulation of the clustering problem that is modeled as an assignment problem of the customers to the vehicles. In particular, we denote by $\mathcal{C}_k$ the set of customers assigned to vehicle $k \in \mathcal{U}$.

The decision variables are defined as follows:

$$y_{ik} = \begin{cases} 1, & \text{if node } i \text{ belongs to cluster } \mathcal{C}_k \\ 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

Moreover, the following auxiliary variables are used:

$$x_{ij} = \begin{cases} 1, & \text{if nodes } i \text{ and } j \text{ are in the same cluster} \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

Now, the time windows are modified by following [70] in order to make them narrower by considering the distance from the first and last depots. More precisely, the time window of node i is updated as follows:

$$[max\{a_0 + \theta_{0i}, a_i\}, min\{b_{C+1} - \theta_{i,C+1}, b_i\}]\,. \tag{4.6}$$

We denote by $R_{ij} = max\{a_i + p_i + t_{ij}, a_j + p_j + t_{j,C+1}\}$ the duration of the travel that starts from node 0, visits nodes i, then j and ends to the final depot $C+1$. The rationale of considering this quantity is that if it holds $R_{ij} > H$ then nodes i and j can not belong to the same cluster.

The clustering phase determines U clusters $\mathcal{C}_k = \{i \in \mathcal{C} | y_{ik} = 1\}$ with $k = 1, ..., U$.

Now, the cost of cluster $\mathcal{C}_k$ is defined by the following index:

$$CC_k = \sum_{i,j \in \mathcal{C}_k} t_{ij} x_{ij}, \tag{4.7}$$

which is the sum of the traveling times between all the pairs of nodes belonging to cluster $\mathcal{C}_k$.

The following ILP problem aims at minimizing the cost of all the clusters:

$$\begin{aligned} &\min CC = \sum_{k \in \mathcal{U}} CC_k \\ &s.t. \end{aligned} \tag{4.8}$$

$$\sum_{k \in \mathcal{U}} y_{ik} = 1, \ \forall \ i \in \mathcal{C} \tag{4.8a}$$

$$\sum_{i \in \mathcal{C}} q_i y_{ik} \leq Q, \ \forall \ k \in \mathcal{U} \tag{4.8b}$$

$$y_{ik} + y_{jk} \leq x_{ij} + 1, \quad \forall \ k \in \mathcal{U}, \ \forall \ i, j \in \mathcal{C}, \ i \neq j \tag{4.8c}$$

$$y_{ik} + y_{jk} \leq 1, \ \forall \ k \in \mathcal{U}, \ \forall \ i, j \in \mathcal{C} : \quad a_i + p_i + t_{ij} \geq b_j \ \wedge \ a_j + p_j + t_{ji} \geq b_i \tag{4.8d}$$

$$y_{ik} + y_{jk} \leq 1, \ \forall \ k \in \mathcal{U}, \ \forall \ i, j \in \mathcal{C} : \quad R_{ij} > b_{C+1} \ \wedge \ R_{ji} > b_{C+1} \tag{4.8e}$$

$$x_{ij} \in \{0, 1\}, \ \forall \ i, j \in \mathcal{C} \tag{4.8f}$$

$$y_{ik} \in \{0, 1\}, \ \forall \ i \in \mathcal{C}, \ \forall \ k \in \mathcal{U} \tag{4.8g}$$

Constraints (4.8a) impose that each node is assigned to only one cluster. Constraints (4.8b) assure that each vehicle is loaded up to not more than its capacity value. Constraints (4.8c) ensure that $x_{ij} = 1$ if and only if i and j are in the same cluster, i.e., $y_{ik} = y_{jk} = 1$. Constraints (4.8d) state that customers i and j can not be in the same cluster due to their time windows. On the other hand, constraints (4.8e) impose that nodes i and j do not belong to the same cluster if $R_{ij} > H$ and $R_{ji} > H$.
About the choice of CC_k as the cost of a cluster, the basic idea is that if the customers belonging to a cluster are close to each other, the corresponding vehicle will perform a short tour into the cluster.

4.3.3 The Distributed Clustering Phase for VRPTW

The problem (4.8) presented in the previous subsection is NP-hard and can not be solved efficiently and in a reasonable time in a centralized way. Therefore, we propose a distributed algorithm that exploits communication between neighbor vehicles and relies

on their capability of solving small instances of the clustering optimization problem. The distributed algorithm relies on the iterative solution of local versions of (4.8) involving a subset of the vehicles in the network.

First, let us define the fully connected communication graph $G_c = (\mathcal{U}, \mathcal{E}_c)$ where the set of nodes $\mathcal{U}$ is the set of vehicles and $\mathcal{E}_c$ is the set of edges, where edge $(k, l) \in \mathcal{E}_c$ describes the communication between vehicles k and l. The graph G_c is introduced for the formulation of the problem and the explanation of the proposed distributed algorithm. Indeed, the graph G_c is useful for: i) enlightening that a communication network among vehicles is supposed to exist; ii) visualizing the concept of neighborhood of vehicles. Moreover, it is worth anticipating that G_c is involved in the proof of the convergence of the proposed algorithm. In addition, since G_c is fully connected, it is implicitly assumed that there is an edge between any pair of vehicles, i.e., there is no restrictions on communication.

At each iteration t of the distributed algorithm, a randomly selected vehicle $k \in \mathcal{U}$ solves a local version of (4.8), denoted with L-ILP, with a subset $\mathcal{NU}_k \subset \mathcal{U}$ of $\delta < U$ randomly selected nodes of G_c. Therefore, the L-ILP at iteration t involves all the vehicles belonging to the set $\mathcal{U}_t = \{k\} \cup \mathcal{NU}_k$, where $k \in \mathcal{U}$ is the vehicle chosen at random at time t (then $|\mathcal{U}_t| = \delta + 1$).

Thus, we denote by $\mathcal{CU}_t \subset \mathcal{C}$ the set of customers assigned to the vehicles belonging to $\mathcal{U}_t$ at time $t - 1$.

The assignment state of vehicle $k \in \mathcal{U}$ at time t is denoted by vector $y_k(t) \in \{0, 1\}^C$ and the cluster assigned to vehicle k at time t is $\mathcal{C}_k(t) = \{i \in \mathcal{CU}_t | y_{ik}(t) = 1\}$. Hence, matrix $Y(t)$ of vectors $y_k(t)$ denotes the overall customer assignment of the vehicles in $\mathcal{U}$.

Moreover, we define:

$$x_{ij}(t) = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are in the same cluster at time } t \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

Hence, the cluster cost at time t is

$$CC_k(t) = \sum_{i,j \in \mathcal{C}_k(t)} t_{ij} x_{ij}(t).$$

In the following, we present the distributed graph partitioning problem:

L-ILP1

$$\min \sum_{i,j \in \mathcal{CU}_t} t_{ij} x_{ij}(t) \tag{4.10}$$

$$s.t.$$

$$\sum_{l \in \mathcal{U}_t} y_{il}(t) = 1, \ \forall \ i \in \mathcal{CU}_t \tag{4.10a}$$

$$\sum_{i \in \mathcal{CU}_t} q_i y_{il}(t) \leq Q, \ \forall \ l \in \mathcal{U}_t \tag{4.10b}$$

$$y_{il}(t) + y_{jl}(t) \leq x_{ij}(t) + 1, \tag{4.10c}$$

$$\forall \ l \in \mathcal{U}_t, \ \forall \ i, j \in \mathcal{CU}_t, \ i \neq j$$

$$y_{il}(t) + y_{jl}(t) \leq 1, \ \forall \ l \in \mathcal{U}_t, \ \forall \ i, j \in \mathcal{CU}_t : \tag{4.10d}$$

$$a_i + p_i + t_{ij} \geq b_j \ \wedge \ a_j + p_j + t_{ji} \geq b_i$$

$$y_{il}(t) + y_{jl}(t) \leq 1, \ \forall \ l \in \mathcal{U}_t, \ \forall \ i, j \in \mathcal{CU}_t : \tag{4.10e}$$

$$R_{ij} > b_{C+1} \ \wedge \ R_{ji} > b_{C+1}$$

$$x_{ij}(t) \in \{0, 1\}, \ \forall \ i, j \in \mathcal{CU}_t \tag{4.10f}$$

$$y_{il}(t) \in \{0, 1\}, \ \forall \ i \in \mathcal{CU}_t, \ \forall \ l \in \mathcal{U}_t \tag{4.10g}$$

If the L-ILP1 problem does not have a feasible solution, then constraints (4.10b), (4.10d) and (4.10e) are relaxed for vehicle k, obtaining the always feasible problem L-ILP2. In other words, vehicle k temporary takes charge of customers whose assignment does not fit all constraints.

L-ILP2

$$\min \sum_{i,j \in \mathcal{CU}_t} t_{ij} x_{ij}(t) \tag{4.11}$$

$$s.t.$$

$$\sum_{l \in \mathcal{U}_t} y_{il}(t) = 1, \ \forall \ i \in \mathcal{CU}_t \tag{4.11a}$$

$$\sum_{i \in \mathcal{CU}_t} q_i y_{il}(t) \leq Q, \ \forall \ l \in \mathcal{NU}_k \tag{4.11b}$$

$$y_{il}(t) + y_{jl}(t) \leq x_{ij}(t) + 1, \tag{4.11c}$$

$$\forall \ l \in \mathcal{U}_t, \ \forall \ i, j \in \mathcal{CU}_t, \ i \neq j$$

$$y_{il}(t) + y_{jl}(t) \leq 1, \ \forall \ l \in \mathcal{NU}_k, \ \forall \ i, j \in \mathcal{CU}_t : \tag{4.11d}$$

$$a_i + p_i + t_{ij} \geq b_j \ \wedge \ a_j + p_j + t_{ji} \geq b_i$$

$$y_{il}(t) + y_{jl}(t) \leq 1, \ \forall \ l \in \mathcal{NU}_k, \ \forall \ i, j \in \mathcal{CU}_t : \tag{4.11e}$$

$$R_{ij} > b_{C+1} \ \wedge \ R_{ji} > b_{C+1}$$

$$x_{ij}(t) \in \{0, 1\}, \ \forall \ i, j \in \mathcal{CU}_t \tag{4.11f}$$

$$y_{il}(t) \in \{0, 1\}, \ \forall \ i \in \mathcal{CU}_t, \ \forall \ l \in \mathcal{U}_t \tag{4.11g}$$

Algorithm 1 shows the steps that the vehicles perform in order to reach a common solution to the clustering problem.

We respectively denote by y_l^* and CC_l^* the solution provided by the L-ILP problems at iteration t and the corresponding cluster cost, both associated to vehicle $l \in \mathcal{U}_t$.

Algorithm 1

Step 1. ***Set*** $t = 0$ *and* ***set*** *the initial assignment configuration* $y_k(0), \ \forall \ k \in \mathcal{U}$.

Select *at random a vehicle* $k \in \mathcal{U}$, *denoted as "master vehicle".*

Set $k_{previous} = \emptyset$

Step 2. ***Set*** $t = t + 1$, ***select*** δ *neighbors at random, including* $k_{previous}$.

Set $\mathcal{CU}_t = \{i \in \mathcal{C} | y_{il}(t-1) = 1 \text{ with } l \in \mathcal{U}_t\}$.

Step 3. ***Solve*** *the L-ILP1 problem,*

If *the L-ILP1 does not have a feasible solution*

then solve *the L-ILP2 problem*

Else

If $\sum_{l\in\mathcal{U}_t} CC_l^* = \sum_{l\in\mathcal{U}_t} CC_l(t-1)$

then set $y_l^* = y_l(t-1)$, $\forall l \in \mathcal{U}_t$

End If

Step 4. ***Set*** $y_l(t) = y_l^*$, $\forall\ l \in \mathcal{U}_t$.

Step 5. ***Select*** *at random a vehicle* $\widehat{k} \in \mathcal{U}$ *as the new "master vehicle",*

Set $k_{previous} = k$ *and* $k = \widehat{k}$.

Go to *Step 2.*

At the beginning of Algorithm 1, a vehicle k and an initial clustering configuration are randomly chosen. Then, at Step 2, vehicle k (the *master vehicle* of the current iteration) selects at random δ neighbors, including the master vehicle of the previous iteration (if $t > 1$). At Step 3 of the algorithm, k solves, with its neighbors, the L-ILP1 problem: if the solution is not feasible, the L-ILP2 is solved with the corresponding relaxations. At step 4, the assignment is updated and, at Step 5, a new *master vehicle* is randomly chosen, the previous k is temporarily stored, and a new iteration can start from Step 2. Now, we prove that Algorithm 1 converges almost surely in finite time to a feasible solution and then it finds a (sub)optimal solution.

Proposition 4.3.1 *Consider a fully connected network of customers G and U vehicles that execute Algorithm 1 in a fully connected communication graph G_c. Let $Y(t)$ be the customer assignment obtained at time t. If the ILP problem (4.8) is feasible, then:*

$$Pr(\exists\tau > 0 : Y(\tau) \quad \textit{is feasible} \quad \forall t \geq \tau) = 1 \tag{4.12}$$

and for each $t > \tau$ the objective function CC associated to $Y(t)$ does not increase.

Proof: In the following, we denote *infeasible* a vehicle $k \in \mathcal{U}$ that violates capacity and time constraints (4.10b), (4.10d) and (4.10e).

At each time $t > 0$, at Step 3 of Algorithm 1 two cases are possible: i) L-IP1 is not feasible and the L-ILP2 is solved; ii) L-IP1 is feasible and is solved.

Let us consider the first iteration $t = 1$ and assume that the initial task assignment $Y(0)$ makes the first randomly chosen vehicle $k \in U$ *infeasible* since it violates capacity and time constraints (4.10b), (4.10d) and (4.10e). If L-IP1 is not feasible, then L-ILP2 is executed. The solution of L-ILP2 can give as a result a new infeasible assignment: the randomly chosen vehicle k is still *infeasible* while all the other vehicles belonging to $\mathcal{NU}_k$ are feasible. However, at time $t + 1$ the number of infeasible vehicles does not increase. In the successive iteration, a neighbour $\overline{k}$ of vehicle k is selected. The new vehicle $\overline{k}$ will solve the local programming problem with its neighbours including vehicle k. Then L-IP1 can be feasible or not feasible. If it is not feasible, then L-ILP2 is solved again and the new selected node $\overline{k}$ of G_c will violate the capacity and time constraints. Hence, at each iteration only one vehicle will be infeasible.

Hence, the *infeasible* node starts at a point of the graph G_c, a neighbour of it is randomly selected, and the infeasible vehicle moves to it; then another neighbor node is selected at random and the infeasible vehicle moves to it etc. The random sequence of infeasible vehicles is a random walk on G_c [72].

If the ILP problem 4.8 is feasible, then the infeasible vehicle performs a random walk on G_c and there is a finite probability that each node of the graph is involved in at least an optimization. Hence, in finite time the customers will be distributed among the vehicles and a feasible solution is found in a finite number τ of iterations. Moreover, for each time $t > \tau$ each vehicle $k \in \mathcal{U}$ is *feasible*: this proves that $Y(\tau)$ is feasible $\forall\ t \geq \tau$.

Moreover, if $Y(t)$ is feasible for $t > \tau$, then only the L-IP1 is solved for $t > \tau$. After solving the L-IP1 at iteration t, two cases are possible: i) the local maximum cost decreases; ii) the local maximum cost stays the same.

Therefore, we can state that, at the end of iteration t, $\sum_{l \in \mathcal{U}_t} CC_l(t) \leq \sum_{l \in \mathcal{U}_t} CC_l(t-1)$ and the local maximum is non-increasing. Hence, also the objective function CC is non-increasing by construction.

Let us consider that the objective function CC is lower bounded by the value of CC^* corresponding to the optimal task assignment Y^*, obtained by solving the optimization problem (4.3). Hence, at each iteration of the algorithm the objective function CC can decrease and the suboptimal solution can improve. □

4.3.4 Traveling Salesman Problem with Time Windows phase

Algorithm 1 provides a set of clusters $\mathcal{C}_k$ and each vehicle $k \in \mathcal{U}$ has a cluster $\mathcal{C}_k$ of customers to serve. Hence, after the clustering phase the TSPTW phase starts.
Let us define $\mathcal{N}_k = \mathcal{C}_k \cup \{0, C+1\} \subset \mathcal{N}$ the set of customers assigned to vehicle $k \in \mathcal{U}$ along with the departure and destination depots.
The decision variables are defined as follows:

$$z_{ij}^k = \begin{cases} 1 & \text{if } k \in \mathcal{U} \text{ visits } j \in \mathcal{N}_k \text{ immediately after } i \in \mathcal{N}_k \\ 0 & \text{otherwise,} \end{cases} \tag{4.13}$$

$$s_{ik} \in \mathbb{R}^+ \text{ start service time of } k \in \mathcal{U} \text{ at node } i \in \mathcal{N}_k. \tag{4.14}$$

The total traveling time of each vehicle $k \in \mathcal{U}$ is defined as follows:

$$TR_k^S = \sum_{i,j \in \mathcal{N}_k} t_{ij} z_{ij}^k. \tag{4.15}$$

Each vehicle $k \in \mathcal{U}$ solves the following TSPTW problem by minimizing its total traveling time:

$$\begin{aligned} &\min TR_k^S \\ &s.t. \end{aligned} \tag{4.16}$$

$$\sum_{j \in \mathcal{N}_k} z_{ij}^k = 1, \ \forall \ i \in \mathcal{C}_k \tag{4.16a}$$

$$\sum_{j \in \mathcal{N}_k} z_{0j}^k = 1, \tag{4.16b}$$

$$\sum_{i \in \mathcal{N}_k} z_{ih}^k - \sum_{j \in \mathcal{N}_k} z_{hj}^k = 0, \ \forall \ h \in \mathcal{C}_k \tag{4.16c}$$

$$\sum_{i \in \mathcal{N}_k} z_{i,C+1}^k = 1 \tag{4.16d}$$

$$z_{ij}^k (s_{ik} + \theta_{ij} - s_{jk}) \leq 0, \ \forall \ i, j \in \mathcal{N}_k \tag{4.16e}$$

$$a_i \leq s_{ik} \leq b_i, \ \forall \ i \in \mathcal{N}_k \tag{4.16f}$$

$$z_{ij}^k \in \{0,1\}, \ \forall \ i,j \in \mathcal{N}_k. \tag{4.16g}$$

Constraints (4.16a) impose that each customer is visited exactly once. Constraints (4.16b), (4.16c) and (4.16d) are flow equations ensuring that the starting and ending nodes of the trip of vehicle k is the depot. Constraints (4.16e) force a vehicle, which is traveling from i to j, to arrive at node j after serving i and traveling between the two nodes. Finally, constraints (4.16f) represent time window constraints for the assigned customers.

Note that, strictly, (4.16) is not a MIP problem because of the non linear restrictions (4.16e), but they can be linearized by the big-M technique as shown for constraints (4.3f).

4.4 The Dynamic Vehicle Routing Problem with Time Windows

In this section we present a MIP formulation of a MDVRPTW with soft time windows constraints that can be applied in a dynamic setting of the problem. Indeed, we assume that, during the vehicle routing, a set of customers requires new services. Hence, the vehicles perform a re-computation of the routing plan. Three main differences are considered: i) the fleet is heterogeneous as vehicle can have different residual capacities Q_k; ii) time windows are soft on early time a_i; iii) the depot set is $\mathcal{D}$ such that $d_k \in \mathcal{D}$ if d_k represents the current position of vehicle k.

In this case the fully connected indirect graph $G = (\mathcal{N}, \mathcal{E})$ describing the network is composed by the set of nodes $\mathcal{N} = \mathcal{C} \cup \mathcal{D} \cup \{N+1\}$, i.e., $\mathcal{N}$ includes the initial positions of vehicles, considered as depots, the customs to be visited and the final depot where the routes go to an end.

More precisely, each vehicle $k \in \mathcal{U}$ starts its trip from $d_k \in \mathcal{D}$, ends it in node $i = N+1$ and visits a sequence of customers $i \in \mathcal{C}$ by satisfying their demand amounts q_i and time windows $[a_i, b_i]$, which are *soft* on early times (a_i). Without loss of generality, we assume

that the start time of the new dynamic phase is $t = 0$ and the plan has to not exceed the planning horizon $[0, H_0]$ with $H_0 < H$.
The following new variables are defined:

$$w_{ik} \in \mathbb{R}^+ \text{ time advance of } k \in \mathcal{U} \text{ at node } i \in \mathcal{C}. \tag{4.17}$$

The problem aims at minimizing the total travel time of the vehicle fleet and the violations of the time windows.
Now, the following MIP problem models the described MDVRPTW with soft TW on the early time:

$$\min \sum_{k\in\mathcal{U}} \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}} t_{ij} z_{ij}^k + \sum_{k\in\mathcal{U}} \sum_{i\in\mathcal{C}} \gamma w_{ik} \tag{4.18}$$

$s.t.$

$$\sum_{k\in\mathcal{U}} \sum_{j\in\mathcal{N}} z_{ij}^k = 1, \ \forall\, i \in \mathcal{C} \cup \mathcal{D} \tag{4.18a}$$

$$\sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{N}} q_i z_{ij}^k \leq Q_k, \ \forall\, k \in \mathcal{U} \tag{4.18b}$$

$$\sum_{j\in\mathcal{C}} \sum_{k\in\mathcal{U}} z_{rj}^k = 1, \ \forall\, r \in \mathcal{D} \tag{4.18c}$$

$$z_{ir}^k = 0, \ \forall\, i \in \mathcal{N}, \forall\, r \in \mathcal{D}, \forall\, k \in \mathcal{U} \tag{4.18d}$$

$$\sum_{i\in\mathcal{N}} z_{ih}^k - \sum_{j\in\mathcal{N}} z_{hj}^k = 0, \ \forall\, h \in \mathcal{C}, \ \forall\, k \in \mathcal{U} \tag{4.18e}$$

$$\sum_{i\in\mathcal{C}\cup\mathcal{D}} z_{i,N+1}^k = 1, \ \forall\, k \in \mathcal{U} \tag{4.18f}$$

$$\sum_{j\in\mathcal{N}} z_{N+1,j}^k = 0, \ \forall\, k \in \mathcal{U} \tag{4.18g}$$

$$z_{ij}^k (s_{ik} + \theta_{ij} - s_{jk}) \leq 0, \ \forall\, i, j \in \mathcal{N}, \ \forall\, k \in \mathcal{U} \tag{4.18h}$$

$$0 \leq s_{ik} \leq b_i, \ \forall\, i \in \mathcal{C} \cup \mathcal{D}, \ \forall\, k \in \mathcal{U} \tag{4.18i}$$

$$0 \leq s_{N+1,k} \leq H_0, \ \forall\, k \in \mathcal{U} \tag{4.18j}$$

$$w_{ik} + s_{ik} \geq a_i, \ \forall\, i \in \mathcal{C}, \ \forall\, k \in \mathcal{U} \tag{4.18k}$$

$$w_{ik} \geq 0,\ \forall\, i \in \mathcal{C},\ \forall\, k \in \mathcal{U} \tag{4.18l}$$

$$z_{ij}^{k} \in \{0,1\},\ \forall\, i,j \in \mathcal{N},\ \forall\, k \in \mathcal{U} \tag{4.18m}$$

The objective function of the MIP problem (4.18) aims at minimizing the total travel time of the vehicle fleet and the total amount of penalties. Constraints (4.18a) state that each node can be visited exactly once. Constraints (4.18b) do not allow any vehicle to be loaded over its capacity. Constraints (4.18c) to (4.18g) are flow equations also ensuring that a vehicle starts its trip from a depot $r \in \mathcal{D}$ and ends it at the final depot $i = N + 1$. A constraint of type (4.18h) forces a vehicle, which has to go from customer i to customer j, to start service at j only after serving i and traveling between the two. Constraints (4.18i) impose time windows restrictions for each customer, while constraints (4.18j) specifically refer to the final depot. Lastly, constraints (4.18k) and (4.18l) concern with violations of time windows: they model $w_{ik} = max\{0, a_i - s_{ik}\}$. Note that for each vehicle the service start variables s_{ik} impose a unique route direction thereby eliminating any subtours [70].
Finally, note that, strictly, (4.18) is not a MIP problem because of the non linear restrictions (4.18h). However, they can be linearized by the big-M technique as shown for constraints (4.3f).

4.5 Distributed Dynamic Vehicle Routing Problem

In this section we propose a heuristic decentralized approach to the MDVRPTW problem based on the *cluster first, route second* method.

4.5.1 The Clustering Phase

The clustering phase is modeled as a graph partitioning problem that considers starting locations of vehicles. Indeed, in this dynamic problem, the cluster $\mathcal{C}_k$ includes the set of customers assigned to vehicle $k \in \mathcal{U}$ and the corresponding starting node $d_k \in \mathcal{D}$. More precisely, clusters contain nodes from $\mathcal{C} \cup \mathcal{D}$, where $i \in \{1, ..., C\} \cup \{C + 1, ..., C + U\}$.

Once a depot has been associated with a vehicle k, we represent it by $d_k \in \mathcal{D}$.
The following ILP problem aims at minimizing the cost of all the clusters:

$$\min \sum_{k \in \mathcal{U}} CC_k \tag{4.19}$$
$$s.t.$$

$$\sum_{k \in \mathcal{U}} y_{ik} = 1, \ \forall \ i \in \mathcal{C} \cup \mathcal{D} \tag{4.19a}$$

$$\sum_{i \in \mathcal{D}} y_{ik} = 1, \ \forall \ k \in \mathcal{U} \tag{4.19b}$$

$$\sum_{i \in \mathcal{C}} q_i y_{ik} \leq Q_k, \ \forall \ k \in \mathcal{U} \tag{4.19c}$$

$$y_{(C+k),k} = 1, \ \forall \ k \in \mathcal{U} \tag{4.19d}$$

$$y_{ik} + y_{jk} \leq x_{ij} + 1, \quad \forall \ k \in \mathcal{U}, \ \forall \ i, j \in \mathcal{C} \cup \mathcal{D}, \ i \neq j \tag{4.19e}$$

$$y_{ik} + y_{jk} \leq 1, \ \forall \ k \in \mathcal{U}, \ \forall \ i, j \in \mathcal{C} : \quad a_i + p_i + t_{ij} \geq b_j \ \wedge \ a_j + p_j + t_{ji} \geq b_i \tag{4.19f}$$

$$x_{ij} \in \{0, 1\}, \ \forall \ i, j \in \mathcal{C} \cup \mathcal{D} \tag{4.19g}$$

$$y_{ik} \in \{0, 1\}, \ \forall \ i \in \mathcal{C} \cup \mathcal{D}, \ \forall \ k \in \mathcal{U} \tag{4.19h}$$

Constraints (4.19a) impose that each node is assigned to only one cluster, while constraints (4.19b) state that a cluster must include one, and only one, depot. Constraints (4.19c) assure that each vehicle is loaded up to not more than its capacity and constraints (4.19d) impose early assignments among depots and vehicles. Constraints (4.19e) ensure that $x_{ij} = 1$ if and only if i and j are in the same cluster. Constraints (4.19f) state that customers i and j can not be in the same cluster due to their time windows.

4.5.2 The Distributed Clustering Phase

Consider again the communication graph $G_c = (\mathcal{U}, \mathcal{E}_c)$.

At each iteration t of the distributed algorithm, a randomly selected vehicle $k \in \mathcal{U}$ solves a local version of (4.19), with a set of δ neighbors denoted by $\mathcal{NU}_k$. Therefore, the L-ILP at iteration t involves the vehicles belonging to the set $\mathcal{U}_t = \{k\} \cup \mathcal{NU}_k$. Moreover, $\mathcal{N}_t = \mathcal{CU}_t \cup \mathcal{D}_t$ is the set of customers to be visited at time t by the vehicles in $\mathcal{U}_t$, including the starting depot of k at time t.

In the following, we present the distributed graph partitioning problem that is the distributed version of (4.19):

L-ILP3

$$\min \sum_{i,j \in \mathcal{N}_t} t_{ij} x_{ij}(t) \tag{4.20}$$

$s.t.$

$$\sum_{l \in \mathcal{U}_t} y_{il}(t) = 1, \ \forall\, i \in \mathcal{N}_t \tag{4.20a}$$

$$\sum_{i \in \mathcal{D}_t} y_{il}(t) = 1, \ \forall\, l \in \mathcal{U}_t \tag{4.20b}$$

$$\sum_{i \in \mathcal{CU}_t} q_i y_{il}(t) \leq Q_l, \ \forall\, l \in \mathcal{U}_t \tag{4.20c}$$

$$y_{(C+l),l}(t) = 1, \ \forall\, l \in \mathcal{U}_t \tag{4.20d}$$

$$y_{il}(t) + y_{jl}(t) \leq x_{ij}(t) + 1, \quad \forall\, l \in \mathcal{U}_t, \ \forall\, i,j \in \mathcal{N}_t, \ i \neq j \tag{4.20e}$$

$$y_{il}(t) + y_{jl}(t) \leq 1, \ \forall\, l \in \mathcal{U}_t, \ \forall\, i,j \in \mathcal{CU}_t : \quad a_i + p_i + t_{ij} \geq b_j \ \wedge \ a_j + p_j + t_{ji} \geq b_i \tag{4.20f}$$

$$x_{ij}(t) \in \{0,1\}, \ \forall\, i,j \in \mathcal{N}_t \tag{4.20g}$$

$$y_{il}(t) \in \{0,1\}, \ \forall\, i \in \mathcal{N}_t, \ \forall\, l \in \mathcal{U}_t \tag{4.20h}$$

If the L-ILP3 problem does not have a feasible solution, constraints (4.20c) and (4.20f) are relaxed, for vehicle k, obtaining the always feasible problem L-ILP4. In other words, vehicle k temporary takes charge of customers whose assignment does not fit all constraints.

L-ILP4

$$\min \sum_{i,j \in \mathcal{N}_t} t_{ij} x_{ij}(t) \tag{4.21}$$

$s.t.$

$$\sum_{l \in \mathcal{U}_t} y_{il}(t) = 1, \ \forall \ i \in \mathcal{N}_t \tag{4.21a}$$

$$\sum_{i \in \mathcal{D}_t} y_{il}(t) = 1, \ \forall \ l \in \mathcal{U}_t \tag{4.21b}$$

$$\sum_{i \in \mathcal{CU}_t} q_i y_{im}(t) \leq Q_l, \ \forall \ l \in \mathcal{NU}_k \tag{4.21c}$$

$$y_{(C+l),l}(t) = 1, \ \forall \ l \in \mathcal{U}_t \tag{4.21d}$$

$$y_{il}(t) + y_{jl}(t) \leq x_{ij}(t) + 1, \tag{4.21e}$$
$$\forall \ l \in \mathcal{U}_t, \ \forall \ i, j \in \mathcal{N}_t, \ i \neq j$$

$$y_{il}(t) + y_{jl}(t) \leq 1, \ \forall \ l \in \mathcal{NU}_k, \ \forall \ i, j \in \mathcal{CU}_t : \tag{4.21f}$$
$$a_i + p_i + t_{ij} \geq b_j \ \wedge \ a_j + p_j + t_{ji} \geq b_i$$

$$x_{ij}(t) \in \{0, 1\}, \ \forall \ i, j \in \mathcal{N}_t \tag{4.21g}$$

$$y_{il}(t) \in \{0, 1\}, \ \forall \ i \in \mathcal{N}_t, \ \forall \ l \in \mathcal{U}_t \tag{4.21h}$$

Algorithm 2 shows the steps that the vehicles perform in order to reach a common solution to the clustering problem.

We respectively denote by y_l^* and CC_l^* the solution provided by the L-ILP problems at iteration t and the corresponding cluster cost, both associated to vehicle $l \in \mathcal{U}_t$.

Algorithm 2

Step 1. ***Set*** $t = 0$ *and* ***set*** *the initial assignment configuration* $y_k(0)$, $\forall\ k \in \mathcal{U}$.
Select *at random a vehicle* $k \in \mathcal{U}$, *denoted as "master vehicle".*
Set $k_{previous} = \emptyset$.

Step 2. ***Set*** $t = t + 1$, ***select*** δ *neighbors at random, including* $k_{previous}$.
Set $\mathcal{N}_t = \{i \in \mathcal{C} \cup \mathcal{D} | y_{il}(t-1) = 1 \text{ with } l \in \mathcal{U}_t\}$.

Step 3. ***Solve*** *the L-ILP3 problem,*
If *the L-ILP3 does not have a feasible solution*
then solve *the L-ILP4 problem*
Else
If $\sum_{l \in \mathcal{U}_t} CC_l^* = \sum_{l \in \mathcal{U}_t} CC_l(t-1)$
then set $y_l^* = y_l(t-1)$, $\forall l \in \mathcal{U}_t$
End If

Step 4. ***Set*** $y_l(t) = y_l^*$, $\forall\ l \in \mathcal{U}_t$.

Step 5. ***Select*** *at random a vehicle* $\widehat{k} \in \mathcal{U}$ *as the new "master vehicle",*
Set $k_{previous} = k$ *and* $k = \widehat{k}$.
Go to *Step 2.*

At the beginning of Algorithm 2, a vehicle and an initial clustering configuration are randomly chosen. Then, at Step 2, the randomly chosen vehicle k selects at random δ neighbors, including the master vehicle of the previous iteration (if $t > 1$). At Step 3 of the algorithm, k solves, with its neighbors, the L-ILP3 problem: if no feasible solution is found, the L-ILP4 is solved with the corresponding relaxations. At Step 4, the assignment is updated. Lastly, at Step 5, a new *master vehicle* is randomly chosen, the previous k is temporarily stored, and a new iteration can start from Step 2
The proof that Algorithm 2 converges almost surely in finite time to a feasible solution and then finds a (sub)optimal solution is very similar to the proof of Proposition 4.3.1.

4.5.3 TSP with soft Time Windows phase

Algorithm 2 provides a set of clusters $\mathcal{C}_k$, each of which contains customers and the starting depot $d_k \in \mathcal{D}$ belonging to vehicle $k \in \mathcal{U}$.
Now, each vehicle k has to solve an instance of TSP with soft Time Windows (TSPSTW) on the set of nodes $\mathcal{N}_k = \mathcal{C}_k \cup \{N+1\} \subset \mathcal{N}$, where $N+1$ is the common final depot and d_k is the starting point. More precisely, we consider soft time windows with respect to the early arrival time only.
The following new variables are defined

$$w_{ik} \in \mathbb{R}^+ \text{ time advance of } k \in \mathcal{U} \text{ at node } i \in \mathcal{C}_k - \{d_k\}. \tag{4.22}$$

The total traveling time of each vehicle $k \in \mathcal{U}$ is defined as follows:

$$TR_k^D = \sum_{i,j \in \mathcal{N}_k} t_{ij} z_{ij}^k, \tag{4.23}$$

while the total penalty for service time advances at customers is defined as follow:

$$PA_k = \sum_{i \in \mathcal{C}_k - \{d_k\}} \gamma w_{ik}. \tag{4.24}$$

Each vehicle $k \in \mathcal{U}$ solves the following TSPSTW problem by minimizing the linear combination of its total traveling time and penalty:

$$\begin{aligned} &\min TR_k^D + PA_k \\ &s.t. \end{aligned} \tag{4.25}$$

$$\sum_{j \in \mathcal{N}_k} z_{ij}^k = 1, \ \forall \ i \in \mathcal{C}_k \tag{4.25a}$$

$$\sum_{j \in \mathcal{N}_k} z_{d_k,j}^k = 1 \tag{4.25b}$$

$$\sum_{i \in \mathcal{N}_k} z_{ih}^k - \sum_{j \in \mathcal{N}_k} z_{hj}^k = 0, \ \forall \ h \in \mathcal{C}_k \tag{4.25c}$$

$$\sum_{i \in \mathcal{C}_k} z^k_{i,N+1} = 1 \tag{4.25d}$$

$$z^k_{ij}(s_{ik} + \theta_{ij} - s_{jk}) \leq 0, \ \forall \ i, j \in \mathcal{N}_k \tag{4.25e}$$

$$0 \leq s_{ik} \leq b_i, \ \forall \ i \in \mathcal{C}_k \tag{4.25f}$$

$$0 \leq s_{N+1,k} \leq H_0 \tag{4.25g}$$

$$w_{ik} + s_{ik} \geq a_i, \ \forall \ i \in \mathcal{C}_k - \{d_k\} \tag{4.25h}$$

$$w_{ik} \geq 0, \ \forall \ i \in \mathcal{C}_k - \{d_k\} \tag{4.25i}$$

$$z^k_{ij} \in \{0, 1\}, \ \forall \ i, j \in \mathcal{N}_k. \tag{4.25j}$$

Constraints (4.25a) impose that each customer is visited exactly once, while constraints (4.25b), (4.25c) and (4.25d) are flow equations. A constraint of type (4.25e) forces a vehicle, which is traveling from i to j, to start service at node j after serving i and traveling between the two nodes. Constraints (4.25f) represent the time window restrictions for the assigned customers, while (4.25g) imposes a horizon time limit to the whole trip. Lastly, constraints (4.25h) and (4.25i) refer to time advances that have to satisfy $w_{ik} = max\{0, a_i - s_{ik}\}$.

Again, note that, strictly, (4.16) is not a MIP problem because of the non linear restrictions (4.25e), but they can be linearized by the big-M technique as shown for constraints (4.3f).

4.6 Results and Discussion

In order to study the performances of the two presented approaches, some benchmark problems are considered in both static and dynamic problems.

The presented results are obtained on a PC equipped with 3.40 GHz AMD A4-5300, 8 GB of memory in single-thread mode and by using Python language scripts with Gurobi optimizer 7.0.1.

4.6.1 Centralized Clustering

Before assessing performances of the distributed approach, this sub section has the aim of showing balancing features of the centralized graph partitioning model (4.8), followed by the routing phase (4.16), which is the starting point of the proposed distributed strategy. The centralized formulation is tested with respect to benchmark problems proposed by Solomon [73], considering the 25-customers instances.

Table 4.1 compares the results obtained by the exact solution of (4.8) and corresponding instances of (4.16) with the solutions of the VRPTW formulation (4.3). More precisely, the first column of Table 4.1 shows the benchmark class used in the comparison, the second column presents the mean execution times (T), in seconds, required for obtaining the solution of (4.8), while the third column shows the mean execution times (T_{opt}) of (4.3). Moreover, the last column reports the percentage ratio of these mean times expressed as $T_{\%} = \frac{T}{T_{opt}} \times 100$.

Table 4.1: *Execution time for the centralized clustering*

Problem Class	T	T_{opt}	$T_{\%}$
C1	23.903	468.938	5.097%
R1	256.377	4216.251	6.081%
RC1	82.627	13429.196	0.645%

The results show that the proposed approach exhibits good performance in terms of execution times.

In the following discussion, we define a set of indices in order to evaluate two main performances of the proposed VRPTW solution method: i) the average gap between the objective function value $F = \sum_{k \in \mathcal{U}} TR_k^S$ obtained by the proposed approach (4.8) and (4.16) and the objective function value F_{opt} obtained by (4.3); ii) the fairness of the proposed approach in terms of balanced vehicle traveling times and assigned total load. In particular the following performance indices are defined:

- $\Delta F_{\%} = \frac{(F - F_{opt})}{F_{opt}} \times 100$ is the average value of the gaps between F and F_{opt};
- σ_{TR} is the percentage ratio between the standard deviation of the traveling times

TR_k^S, obtained by the proposed static approach (4.8) and (4.16), and their mean value;

- $\sigma_{TR_{opt}}$ is the percentage ratio between the standard deviation of the traveling times obtained by (4.3) and their mean value;
- σ_L is the percentage ratio between the standard deviation of the load L_k assigned to each vehicle $k \in \mathcal{U}$, obtained by the proposed static approach, and their mean value. Such loads are computed as follows: $L_k = \sum_{i \in \mathcal{C}_k} q_i$;
- $\sigma_{L_{opt}}$ is the percentage ratio between the standard deviation of loads assigned to vehicles by (4.3) and their mean value.

Note that in all the previous cases, the standard deviation is computed by using the corrected sample standard deviation formula, after applying the Bessel's correction [74].

Table 4.2: *Performance Indices with Centralized Clustering*

Problem class	$\Delta F_{\%}$	$\sigma_{TR\%}$	$\sigma_{TR_{opt\%}}$	$\sigma_{L\%}$	$\sigma_{L_{opt}\%}$
C1	0.19%	47.07%	47.18%	26.36%	26.70%
R1	8.50%	13.64%	17.62%	22.93%	28.99%
RC1	0.76%	9.03%	8.39%	10.75%	10.42%

Table 4.2 reports the average values of the defined performance indices for each problem class.

In the worst case of problem class R1, the objective function gap is 8.50% greater than the best known solution, obtained by (4.3), while the other benchmarks show good results. However, it is important to highlight that class R1 has much better results in terms of standard deviation of both the total traveling time and assigned load. This ensures that the vehicles belonging to the fleet have a balanced workloads to perform.

4.6.2 Static Problem

In this subsection, we solve some benchmark problems proposed by Solomon [73], considering 50-customers instances. Since model 4.3 can not be solved in reasonable time, we refer to solution provided by [75].

We call Algo1_TSP the distributed proposed approach that consists of the application of Algorithm 1 and the subsequent decentralized TSPTW phase (4.16) (Levels 1 and 2 of the static problem in Fig. 4.1). In particular, Algorithm 1 goes to an end when for 500 iterations the overall cluster assignment to vehicles does not change.
The obtained solutions provide the overall travel time F^S defined as follows:

$$F^S = \sum_{k \in \mathcal{U}} TR_k^S.$$

We define a set of indices in order to evaluate two main performances of the proposed VRPTW solution method: i) the average gap between the objective function value F^S obtained by Algo1_TSP and the objective function value F^S_{opt} of solution (4.3); ii) the fairness of the proposed approach in terms of balanced vehicle traveling times and assigned total load.
In particular, the following performance indices are defined:

- $\Delta F_{\%} = \frac{(F^S - F^S_{opt})}{F^S_{opt}} \times 100$, the average gap between F^S and F^S_{opt};
- $\sigma_{TR\%}$, the percentage ratio between the standard deviation of traveling times TR_k^S and their mean value;
- $\sigma_{TR^S_{opt}\%}$, the percentage ratio between the standard deviation of traveling times for the exact solutions and their mean value;
- $\sigma_{L\%}$, the percentage ratio between the standard deviation of the loads L_k assigned to each vehicle $k \in \mathcal{U}$, obtained by Algo1_TSP ($L_k = \sum_{i \in \mathcal{C}_k} q_i$), and the mean value of the loads;
- $\sigma_{L_{opt}\%}$, the percentage ratio between the standard deviation of the loads assigned to vehicles in the solution of (4.3) and the mean value of the loads.

Note that in all the previous cases, standard deviations are computed by using the corrected sample standard deviation formula, after applying the Bessel's correction [74]. Table 4.3 reports the average values of the defined performance indices for each problem class and the average values of the selected δ.

Table 4.3: *Performance Indices of the Static Phase*

Problem class	$\Delta F_{\%}$	$\sigma_{TR\%}$	$\sigma_{TR_{opt}\%}$	$\sigma_{L\%}$	$\sigma_{L_{opt}\%}$	δ
C1	0.13%	30.56%	30.60	13.88%	13.88%	1.22
R1	12.38%	20.41%	25.80%	29.53%	41.00%	2.17
RC1	7.36%	12.09%	20.26%	30.65%	33.44%	1.63

The results of the objective function gaps point out that, in the worst case, the proposed approach yields a mean value which is 12.38% greater than the best known solution for the R1 problem class. The other benchmarks show very good results with gaps under the 10%.

However, it is important to highlight that problems with the worst results in terms of $\Delta F_{\%}$, i.e., R1 and RC1, have much better results in terms of standard deviation of both the total traveling time and assigned load. This ensures that in each case the vehicles belonging to the fleet have a balanced work to perform. As a consequence, the proposed solution methodology can be successfully applied in real life scenarios that typically require guaranteeing fair working conditions for the vehicle drivers.

4.6.3 Dynamic Problem

In order to show the performances of the methodology presented in Section 4.5, we have adapted some benchmark problems proposed by Solomon [73] to the multi-depot VRPTW (Appendix A).

Table 4.4 presents the attributes that are significant for the distributed approach, with reference to the 25-customers instances. The first column shows the instance names, the second column reports $U = D$ and the third column shows the values of δ_{max}, which is the highest value assigned to δ.

We call Algo2_TSP the distributed proposed approach that consists of Algorithm 2 and the subsequent decentralized TSPSTW phase (4.25) (Levels 1 and 2 of the dynamic problem in Fig. 4.1). Each performance index is obtained by averaging on values of 5 executions for each benchmark instance. Also in this case, Algorithm 2 stops when for

Table 4.4: *Multi-depot Benchmarks*

Benchmark	D	δ_{max}
MD_C101	3	1
MD_C102	3	1
MD_C103	3	1
MD_C104	3	1
MD_C105	3	1
MD_C106	3	1
MD_C107	3	1
MD_C108	3	1
MD_C109	3	1
MD_R101	8	3
MD_R102	8	3
MD_R103	5	3
MD_R104	4	2
MD_R105	6	3
MD_R106	5	3
MD_R107	4	2
MD_R108	4	2
MD_R109	5	3
MD_R110	5	3
MD_R111	5	3
MD_R112	4	2
MD_RC101	5	3
MD_RC102	3	1
MD_RC103	3	1
MD_RC104	3	1
MD_RC105	4	2
MD_RC106	3	1
MD_RC107	3	1
MD_RC108	3	1

Table 4.5: *Computation times*

Problem class	δ	I	T	T_{opt}	$T_{\%}$
MD_C1	1	6.45	2.263	95.315	2.37%
MD_R1	1	23.22	3.065	4262.915	0.07%
	2	9.6	28.893	4262.915	0.68%
	3	5.56	41.816	4561.658	0.92%
MD_RC1	1	9.271	1.912	3009.373	0.06%
	2	5.25	10.038	1126.766	0.89%
	3	3.5	15.47	106.192	14.57%

500 iterations the overall cluster assignment to vehicles does not change.

Table 4.5 compares execution times obtained by the proposed distributed approach with those obtained by the MIP formulation (4.18). More precisely, the first column of Table 4.5 shows the benchmark class; the second column shows the value of δ; the third column illustrates the mean number of iterations I that Algorithm 2 requires to reach the final assignment; the fourth column presents the mean execution times, T, in seconds required to obtain the solution by Algo2_TSP and the fifth column shows the mean execution time T_{opt} of (4.18). Finally, the last column reports the percentage ratio of these mean times expressed as $T_{\%} = \frac{T}{T_{opt}} \times 100$.

The results show that the proposed approach has very good performance in terms of execution times.

In order to continue comparisons, we remind that the values of the objective functions provided by Algo2_TSP are the following:

$$TR^D = \sum_{k \in \mathcal{U}} TR_k^D,$$

$$PA = \sum_{k \in \mathcal{U}} PA_k,$$

$$F^D = TR^D + PA.$$

In the following discussion, we define a set of indices in order to evaluate four main per-

formances of the proposed MDVRPTW solution method: i) the average gap between F^D and the objective function value F^D_{opt} obtained by solving problem (4.18); ii) the average gap between TR^D and the effective traveling time TR_{opt} obtained by solving problem (4.18); iii) the average time advance W and average number of involved customers C_W; iv) the fairness of the proposed approach in terms of balanced vehicle traveling times and assigned total load.

In particular the following performance indices are defined:

- $\Delta F_{\%} = \frac{(F^D - F^D_{opt})}{F^D_{opt}} \times 100$;
- $\Delta TR_{\%} = \frac{(TR^D - TR^D_{opt})}{TR^D_{opt}} \times 100$ is the percentage gap between TR and TR_{opt};
- W is the average time advance;
- C_W is the number of customers where time window violations take place;
- $\sigma_{TR\%}$ is the percentage ratio between the standard deviation of the traveling times TR^D_k and their mean value;
- $\sigma_{TR_{opt}\%}$ is the percentage ratio between the standard deviation of traveling times for the solution of (4.18) and their mean value;
- $\sigma_{L\%}$ is the percentage ratio between the standard deviation of the load $L_k = \sum_{i \in \mathcal{C}_k} q_i$ assigned to each vehicle $k \in \mathcal{U}$, obtained by Algo2_TSP, and the mean value of loads;
- $\sigma_{L_{opt}\%}$ is the percentage ratio between the standard deviation of the loads assigned to vehicle in the exact solution and their mean value.

Table 4.6 and 4.7 report the average values of the defined performance indices for each problem class.

The results of the objective function gaps $\Delta F_{\%}$ point out that, in the worst case, the proposed approach yields a mean value which is 56.71% greater than the exact solution, obtained by (4.18), for the MD_R1 problem class. This is due to the high value of the

Table 4.6: *Performance Indices (I)*

Problem class	δ	$\Delta F_{\%}$	$\Delta TR_{\%}$	W	C_W
MD_C1	1	1.272%	1.059%	0	0
MD_R1	1	59.383%	11.743%	4.85	0.703
	2	42.601%	8.618%	3.46	0.653
	3	51.080%	9.293%	5.56	0.75
MD_RC1	1	1.842%	1.842%	0	0
	2	1.591%	1.591%	0	0
	3	2.697%	2.697%	0	0

Table 4.7: *Performance Indices (II)*

Problem class	δ	$\sigma_{TR\%}$	$\sigma_{TR_{opt}\%}$	$\sigma_{L\%}$	$\sigma_{L_{opt}\%}$
MD_C1	1	35.49%	35.47%	26.36%	26.36%
MD_R1	1	18.08%	24.86%	23.9%	26.27%
	2	16.64%	24.86%	18.59%	26.27%
	3	20.19%	25.55%	17.98%	28.58%
MD_RC1	1	17.51%	17.78%	13.6%	14.43%
	2	18.98%	21.42%	36.16%	41.06%
	3	11%	14.4%	37.84%	44.6%

penalization parameter $\gamma = 40$, calibrated in order to prevent time windows violations as much as possible. The other benchmarks show very good results.

However, it is important to highlight that the problem class with the worst results in terms of $\Delta F_{\%}$, i.e., MD_R1, exhibits better standard deviation of both the total traveling time and assigned load. This ensures that in each case the vehicles belonging to the fleet have a balanced work to perform.

4.7 Example

In this section, the proposed algorithms are assessed considering a dynamic vehicle routing application inspired by a transport company and built on the basis of Solomon's

benchmarks [73]. This company, acting also as courier service operator, collects packages from customer locations to a central depot while service requests can be either arranged in advance or received and planned in real time. The former requests are included in an initial routing plan that is computed before the start of the working day, when the empty vehicles are still at the central depot, at time $\tau = 0$.

In order to manage ongoing requests and adapt the routing plan, it is necessary to run Algorithm 2 starting from time $\tau = \tau_1$. At that time, each vehicle knows: i) its position and status (it is traveling, waiting at a customer or servicing a customer); ii) its residual capacity; iii) the customers it has already served.

In addition, for the initialization of Algorithm 2, each vehicle initially maintains the already known customers it was assigned in the previous plan and that have not been served yet. On the basis of the updated data, the vehicles firstly perform the distributed multi-depot clustering phase in which every vehicle communicates with neighbors. Secondly, each vehicle solves the TSPSTW (4.25) in which the starting point is its position in $\tau = \tau_1$, while the ending point is the central common depot.

Let us consider a particular instance of the problem. Before the service starts, there are $C = 78$ customer requests to be managed by a fleet of $U = 14$ vehicles still parked at the central depot. Table 4.8 shows, for each node $i \in \mathcal{N}$, the corresponding demand q_i and time window $[a_i, b_i]$, respectively expressed in packaging unit (PU) and minutes. The first and the last entries represent the central depot ($i = 0$ and $i = 79$). In addition, every customer requires a service time $p_i = 10$ minutes, with $i = 1, ..., 78$, and each vehicle has a capacity of $Q = 200$ PUs.

Table 4.8: *Initial Nodes*

i	q_i	a_i	b_i		i	q_i	a_i	b_i
0	0	0	230	—	40	10	0	193
1	10	0	204	—	41	9	0	208
2	13	0	197	—	42	14	95	105
3	19	149	159	—	43	18	0	197
4	3	99	109	—	44	2	136	146
5	5	0	198	—	45	6	130	140
6	9	95	105	—	46	7	0	196

(Continued on next page)

Table 4.8 – *(continued from previous page)*

i	q_i	a_i	b_i		i	q_i	a_i	b_i
7	16	97	107	—	47	28	0	202
8	16	124	134	—	48	13	0	194
9	12	67	77	—	49	19	58	68
10	20	0	187	—	50	10	0	185
11	8	61	71	—	51	9	73	83
12	19	0	190	—	52	20	51	61
13	2	157	167	—	53	25	127	137
14	17	0	187	—	54	25	83	93
15	9	0	188	—	55	6	50	60
16	18	97	107	—	56	15	0	180
17	29	68	78	—	57	25	0	197
18	3	0	190	—	58	9	0	199
19	17	0	208	—	59	8	149	159
20	16	37	47	—	60	18	0	192
21	16	0	213	—	61	13	73	83
22	21	71	81	—	62	3	96	106
23	23	0	186	—	63	23	92	102
24	11	37	47	—	64	6	182	192
25	14	0	183	—	65	16	0	196
26	5	41	51	—	66	11	0	198
27	8	0	198	—	67	7	101	111
28	16	83	93	—	68	41	0	196
29	31	44	54	—	69	35	0	184
30	9	85	95	—	70	26	93	103
31	5	97	107	—	71	9	74	84
32	5	31	41	—	72	3	0	187
33	7	0	185	—	73	2	18	28
34	18	69	79	—	74	27	0	207
35	16	32	42	—	75	20	0	205
36	27	0	185	—	76	12	0	202
37	36	0	192	—	77	10	0	198
38	30	108	118	—	78	9	83	93
39	13	0	203	—	79	0	0	230

On the basis of these data, Algorithm 1 and the subsequent TSPTW phase (4.16) provide the initial routing plan shown in Table 4.9. Clusters are also illustrated in Fig. 4.2, in which the central red star represents the common departure depot.

For the sake of brevity, Table 4.9 gives information about the execution status of the initial plan at time $\tau = \tau_1$, with $\tau_1 = 60$ minutes: customers already served are written in italic, while customers under service are marked with an asterisk. In particular, 19

Table 4.9: *Initial routing plan*

k	Route	Traveling time	Load
1	0 - *66* - *35* - 6 - 65 - 37 - 5 - 79	175.281	93
2	0 - *26* - 51 - 38 - 36 - 14 - 79	190.24	88
3	0 - *10* - 34 - 28 - 69 - 12 - 79	168.33	108
4	0 - *39* - 61 - 63 - 2 - 18 - 64 - 79	213.09	71
5	0 - *41* - *1* - 55* - 71 - 4 - 79	127.2	37
6	0 - *56* - 52 - 7 - 53 - 15 - 79	197.57	85
7	0 - *73* - *47* - *77* - 27* - 70 - 76 - 75 - 74 - 79	165.44	133
8	0 - *21* - *20* - 19 - 30 - 42 - 79	119.47	72
9	0 - *43* - *29* - 54 - 44 - 3 - 79	189.54	95
10	0 - *24* - 40* - 22 - 62 - 25 - 79	166.96	59
11	0 - 49* - 9 - 50 - 8 - 72 - 23 - 79	199.54	83
12	0 - 78 - 67 - 13 - 48 - 68 - 79	223.71	72
13	0 - *32* - *33* - 11 - 31 - 46 - 79	151.37	32
14	0 - *57* - 17 - 16 - 60 - 45 - 59 - 58 - - 79	193.6	113

customer requests are completely served, while 4 are under service.
During the time period $[0, \tau_1]$ other 22 customer requests have arisen, so the new planning phase has to consider:

- a set of customers composed by the not yet served customers, the under-service customers and the new 22 customer requests, whose indices starts from 79 (the new set $\mathcal{C}$ is shown in Table 4.10);
- a set $\mathcal{D} = \{d_1, ..., d_{14}\}$ of $D = 14$ virtual depots;
- residual capacities of the vehicles (Table 4.11);
- a common return central depot that coincides with node $i = 0$.

Table 4.10: *The New Customers Set*

i	q_i	a_i	b_i		i	q_i	a_i	b_i
2	13	0	197	—	58	9	0	199

(Continued on next page)

Table 4.10 – *(continued from previous page)*

i	q_i	a_i	b_i		i	q_i	a_i	b_i
3	19	149	159	—	59	8	149	159
4	3	99	109	—	60	18	0	192
5	5	0	198	—	61	13	73	83
6	9	95	105	—	62	3	96	106
7	16	97	107	—	63	23	92	102
8	16	124	134	—	64	6	182	192
9	12	67	77	—	65	16	0	196
11	8	61	71	—	67	7	101	111
12	19	0	190	—	68	41	0	196
13	2	157	167	—	69	35	0	184
14	17	0	187	—	70	26	93	103
15	9	0	188	—	71	9	74	84
16	18	97	107	—	72	3	0	187
17	29	68	78	—	74	27	0	207
18	3	0	190	—	75	20	0	205
19	17	0	208	—	76	12	0	202
22	21	71	81	—	78	9	83	93
23	23	0	186	—	79	7	60	202
25	14	0	183	—	80	26	60	199
27	8	0	198	—	81	19	60	205
28	16	83	93	—	82	23	159	169
30	9	85	95	—	83	12	60	204
31	5	97	107	—	84	11	60	201
34	18	69	79	—	85	6	172	182
36	27	0	185	—	86	9	60	190
37	36	0	192	—	87	27	60	202
38	30	108	118	—	88	8	143	153
40	10	0	193	—	89	1	60	184
42	14	95	105	—	90	18	200	210
44	2	136	146	—	91	3	60	201
45	6	130	140	—	92	36	142	152
46	7	0	196	—	93	5	182	192
48	13	0	194	—	94	14	179	189
49	19	58	68	—	95	26	60	192
50	10	0	185	—	96	15	60	211
51	9	73	83	—	97	1	60	194
52	20	51	61	—	98	22	188	198
53	25	127	137	—	99	11	60	204
54	25	83	93	—	100	17	185	195
55	6	50	60					

On the basis of the updated data, Algo2_TSP provides the solution shown in Table 4.12.

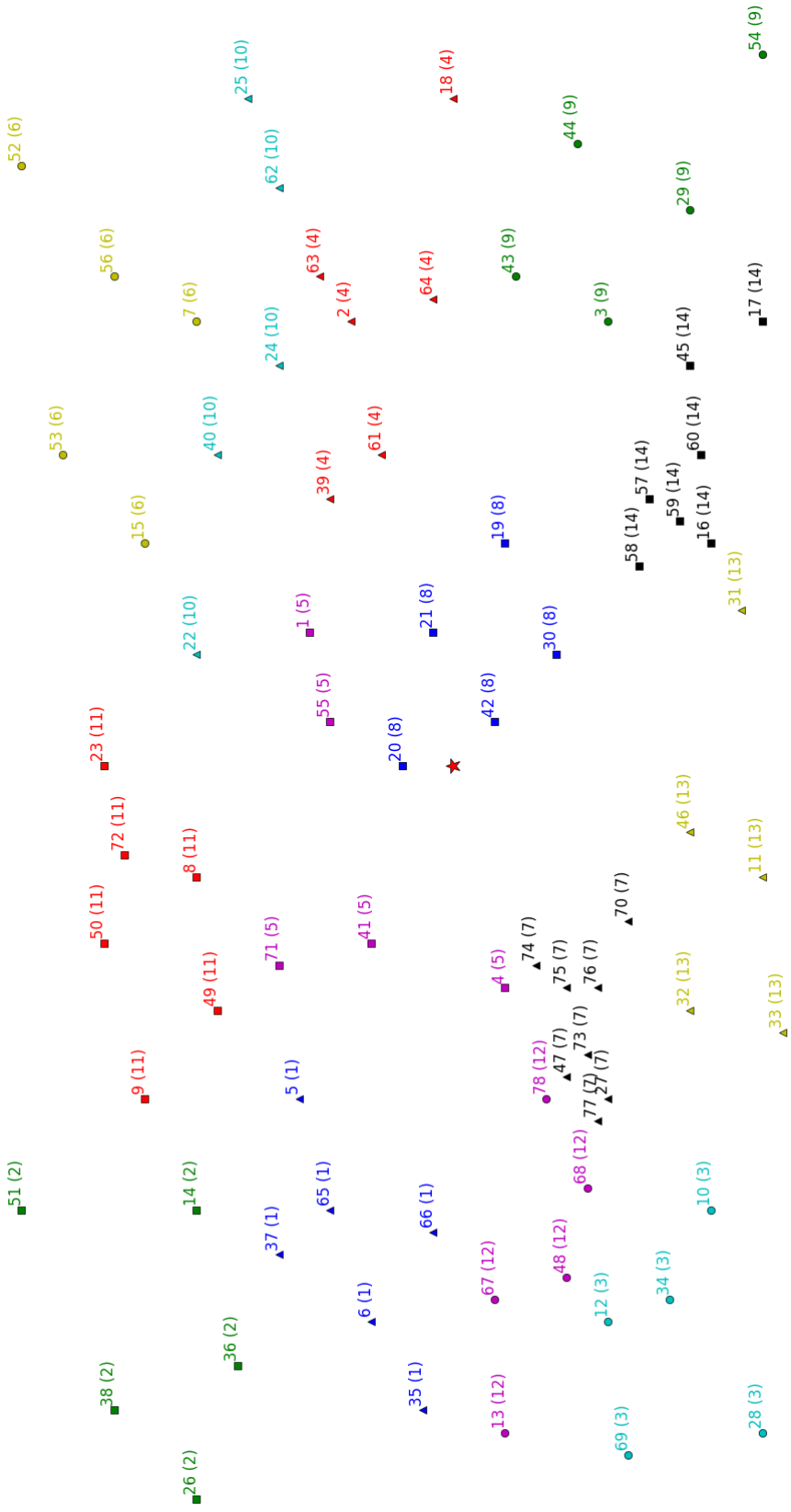

Figure 4.2: *Initial clusters.*

Table 4.11: *Residual Capacities of Vehicles*

k	Q_k		k	Q_k
1	173	—	8	168
2	195	—	9	151
3	180	—	10	189
4	187	—	11	200
5	181	—	12	200
6	185	—	13	188
7	160	—	14	175

Fig. 4.3 shows the updated clusters in which stars represents virtual depots while the central black circle corresponds to the common return depot.

Finally, Table 4.13 shows performances of the static and dynamic problems. Algo2_2TSP (dynamic phase) converges in less iterations than Algo1_TSP (static phase) because the former starts from an initial assignment which is, for most of it, inherited from the latter, since only 22 customers on 100 are new and randomly assigned at first. About sizes of neighborhoods, the static phase is solved with $\delta = 2$ because it gives a more reproducible solution than $\delta = 1$. In the dynamic phase, instead, $\delta = 1$ and $\delta = 2$ lead to the same solution. The reason is the same as for the minor number of iterations. Moreover, both solutions do not need time windows violations.

Table 4.12: *Updated Routing Plan*

k	Route	Traveling time	Load
1	d_1 - 6 - 89 - 37 - 65 - 5 - 83 - 0	58.166	79/173
2	d_2 - 51 - 38 - 36 - 14 - 0	78.301	83/195
3	d_3 - 34 - 28 - 69 - 12 - 97 - 100 - 0	62.483	106/180
4	d_4 - 61 - 63 - 86 - 18 - 92 - 2 - 64 - - 0	71.38	103/187
5	d_5 - 55 - 71 - 4 - 96 - 87 - 93 - 0	85.231	65/181
6	d_6 - 52 - 62 - 25 - 88 - 94 - 0	84.082	59/185
7	d_7 - 27 - 70 - 76 - 75 - 99 - 74 - 82 - - 0	39.384	127/160
8	d_8 - 84 - 30 - 42 - 81 - 19 - 90 - 0	64.186	88/168
9	d_9 - 54 - 44 - 3 - 85 - 0	80.424	52/151
10	d_{10} - 40 - 22 - 7 - 53 - 15 - 95 - 0	89.934	107/189
11	d_{11} - 49 - 9 - 50 - 8 - 72 - 23 - 0	71.155	83/200
12	d_{12} - 68 - 78 - 67 - 91 - 80 - 13 - 48 - - 98 - 0	78.258	123/200
13	d_{13} - 11 - 31 - 46 - 79 - 0	54.457	27/188
14	d_{14} - 17 - 16 - 60 - 45 - 59 - 58 - 0	51.094	88/175

Table 4.13: *Performance Indices of the Example*

Phase	δ	I	T	$\sigma_{TR\%}$	$\sigma_{L\%}$
Static	2	56	270.6	26.2%	33.9%
Dynamic	2	29	114.2	21.3%	32.7%
	1	52	12.2	21.3%	32.7%

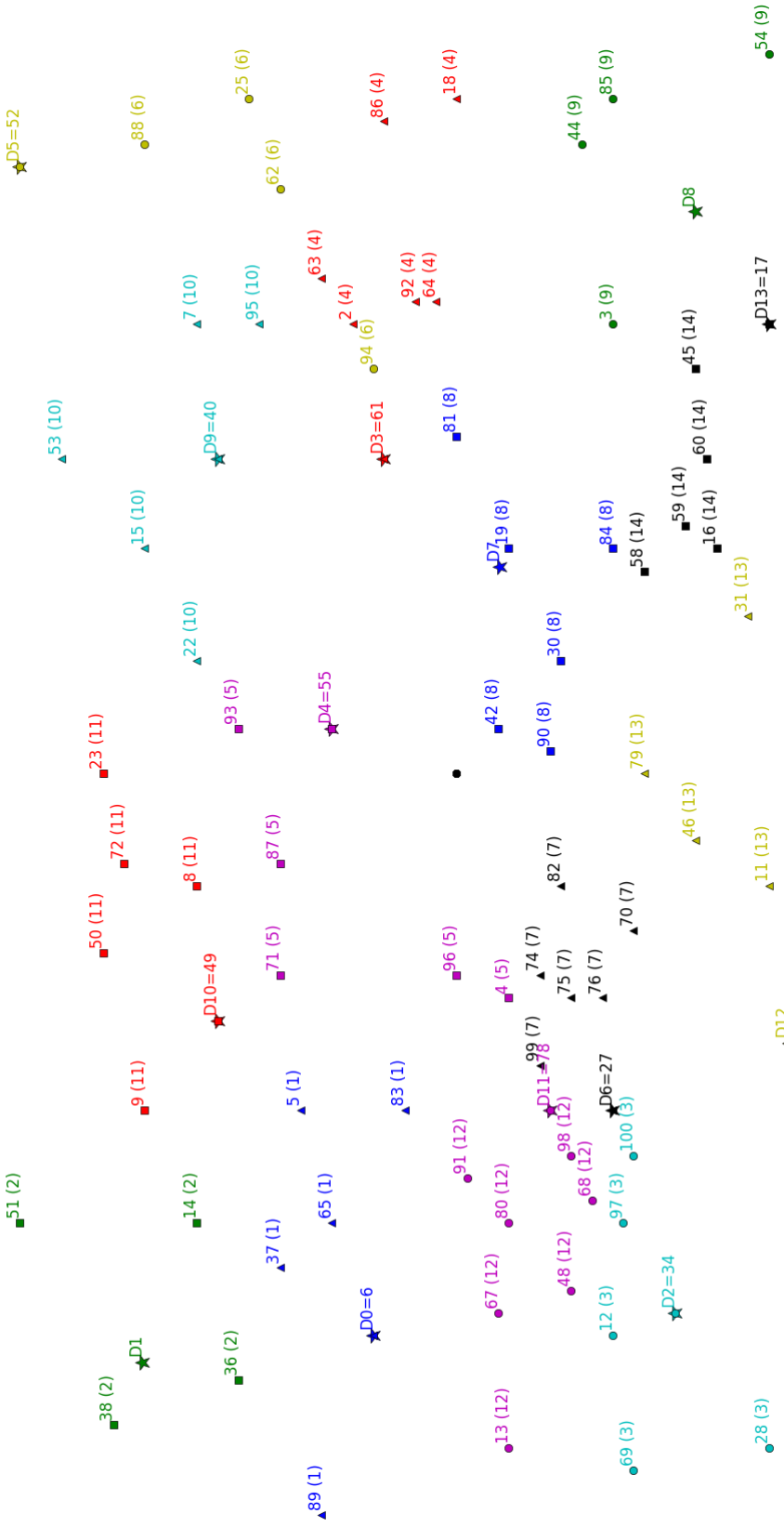

Figure 4.3: *Updated clusters.*

Chapter 5

Conclusion

This thesis faces various aspects of the VRP which are consequences of new domain requirements, such as timeliness and attention to different stakeholders, and of technological developments, especially in ICTs. Among such consequences, there are: i) the need of VRP formulations able to interact with dynamic environments, i.e., DVRP, and the opportunity of designing distributed approaches for both static and dynamic settings; ii) the possibility, which is also a necessity for companies, of inserting vehicle routing planner modules into ICT systems such as DSSs.

Firstly, the thesis analyzes recent literature about DVRP, enlightening taxonomy and classifications by source of dynamism, applications and methodologies. The analysis points out that the most considered *source of dynamism* is the arrival of ongoing service requests, while *dynamic-deterministic* problems and centralized approaches are still the most studied in papers.

Secondly, the thesis presents an architecture for a DSS that includes a VRP module devoted to critical services in city logistics, such as waste collection. The aim is to support decision makers in defining work shifts by minimizing number of shifts and overall travel times. In order to deal with large real cases, a *cluster first, route second* heuristic algorithm is proposed, where clusters are built starting from the farthest not assigned node and the routing stage is solved by the farthest insertion heuristic.

The algorithm is validated by comparisons with specific MILP formulations for the Postal

Delivery and Waste Collection services. In particular, for the Postal Delivery service, a heterogeneous vehicle fleet is involved and a post-fit procedure is exploited in order to select the most suitable vehicle for each cluster of delivery points. About the Waste Collection service, a complex and constrained problem is addressed: i) capacity and time constraints are taken into account; ii) a network with a large number of bins characterized by different demand and pick up times is considered; iii) each vehicle can perform several routes during a daily shift; iv) different trip speeds are considered for different phases of the collection service. Then, both services are assessed by real case studies, with good results in term of effectiveness and computation time: i) for the Postal Delivery service, the Apulian distribution network, with center in the city of Bari, Southern Italy, is considered; ii) for the Waste Collection service, the case of the city of Trieste, Northern Italy, is considered.

Finally, this thesis proposes a distributed approach for the VRPTW and the MD-VRPTW, which can act as components of a DVRP, in which the source of dynamism is the arrival of new service requests. The general strategy is the *cluster first, route second* and the core of the approach consists of an asynchronous, randomized and distributed algorithm. More precisely, vehicles, which are considered as autonomous intelligent agents, reach the final assignment by iteratively solving local Graph Partitioning problems, in the form of Local-Integer Linear Programming problems, with randomly selected neighbor vehicles. Afterwards, each vehicle can optimize the route into its own cluster, by solving a small instance of the TSPTW.

The proposed approach is assessed for both VRPTW and MDVRPTW by comparisons with exact and centralized approaches. Good results are obtained with regard to balanced workloads in terms of average traveling times, average vehicle loads, and the corresponding standard deviations. Moreover, an example inspired by a transport company shows the applicability of the proposed approach in real-world scenarios.

About future work, there are numerous open problems for further research:

- compatibly with computation time requirements, enhancing the quality of solutions provided by the proposed algorithms by including the clustering and the

routing phases into an iterative scheme, which exploits the result of the routing for redefining the clustering;

- merging between the proposed DSS and distributed approach for both static and dynamic settings;
- considering other sources of dynamism such as vehicle breakdowns and communication losses among vehicles;
- considering stochastic aspects in the whole decision process in order to capture uncertainty and forecast the demand and traffic factors;
- deepening communication and coordination aspects among vehicles in order to propose a parallel distributed approach, which requires a way to avoid overlaps among different neighborhoods of vehicles;
- investigating on other distributed strategies that involve metaheuristics.

Appendices

Appendix A

Benchmarks

In order to show the performances of the methodology presented in Section 4.5, we have adapted some benchmark problems proposed by Solomon [73] to the considered multi-depot VRPTW. In particular, 25-customers instances of types *C1* (clustered positions), *R1* (random positions), and *RC1* (mixed positions) are considered and, for each instance, the section devoted to nodes is extended with positions of vehicles, i.e., with depots whose indices start from 26. Nodes from 1 to 25 represent the same customers as in Solomon's benchmarks, while node with index 0 is the final common return depot, which is denoted by $i = N + 1$ in Section 4.5.

Appendix A continues with three sections that contain the benchmarks derived from class *C1*, class *R1* and class *RC1* [73], respectively. The new classes are denoted by *MD_C*, *MD_R* and *MD_RC*

Finally, each benchmark instance is shown in a table with a two-lines heading: the first one indicates the name of the instance and the number D of departing depots or vehicles; the second one concerns characteristics of each node:

- id is the progressive number;
- the couple (x, y) denotes the position in a common reference system;
- q represents the demand;
- a and b define, respectively, the early time and the late time of the time window;

- p_i denotes the service time.

The capacity of vehicles is $Q = 200$.

A.1 Benchmarks MD_C

Table A.1: *Benchmark MD_C101*

MD_C101				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	1236	0
1	45	68	10	912	967	90
2	45	70	30	825	870	90
3	42	66	10	65	146	90
4	42	68	10	727	782	90
5	42	65	10	15	67	90
6	40	69	20	621	702	90
7	40	66	20	170	225	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	652	721	90
13	22	75	30	30	92	90
14	22	85	10	567	620	90
15	20	80	40	384	429	90
16	20	85	40	475	528	90
17	18	75	20	99	148	90
18	15	75	20	179	254	90
19	15	80	10	278	345	90
20	30	50	10	10	73	90
21	30	52	20	914	965	90
22	28	52	20	812	883	90
23	28	55	10	732	777	90
24	25	50	10	65	144	90
25	25	52	40	169	224	90
26	40	50	0	0	1236	0
27	15	65	0	0	1236	0
28	45	85	0	0	1236	0

Table A.2: *Benchmark MD_C102*

MD_C102				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	1236	0
1	45	68	10	0	1127	90
2	45	70	30	0	1125	90
3	42	66	10	0	1129	90
4	42	68	10	727	782	90
5	42	65	10	0	1130	90
6	40	69	20	621	702	90
7	40	66	20	0	1130	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	0	1107	90
13	22	75	30	30	92	90
14	22	85	10	567	620	90
15	20	80	40	384	429	90
16	20	85	40	475	528	90
17	18	75	20	99	148	90
18	15	75	20	179	254	90
19	15	80	10	278	345	90
20	30	50	10	10	73	90
21	30	52	20	0	1135	90
22	28	52	20	812	883	90
23	28	55	10	732	777	90
24	25	50	10	65	144	90
25	25	52	40	169	224	90
26	40	50	0	0	1236	0
27	15	65	0	0	1236	0
28	45	85	0	0	1236	0

Table A.3: *Benchmark MD_C103*

MD_C103				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	1236	0
1	45	68	10	0	1127	90
2	45	70	30	0	1125	90
3	42	66	10	0	1129	90

(Continued on next page)

Table A.3 – *(continued from previous page)*

MD_C103				$D = 3$		
id	x	y	q	a	b	p
4	42	68	10	727	782	90
5	42	65	10	0	1130	90
6	40	69	20	621	702	90
7	40	66	20	0	1130	90
8	38	68	20	255	324	90
9	38	70	10	534	605	90
10	35	66	10	357	410	90
11	35	69	10	448	505	90
12	25	85	20	0	1107	90
13	22	75	30	30	92	90
14	22	85	10	0	1106	90
15	20	80	40	384	429	90
16	20	85	40	0	1105	90
17	18	75	20	99	148	90
18	15	75	20	0	1110	90
19	15	80	10	0	1106	90
20	30	50	10	0	1136	90
21	30	52	20	0	1135	90
22	28	52	20	812	883	90
23	28	55	10	732	777	90
24	25	50	10	0	1131	90
25	25	52	40	169	224	90
26	40	50	0	0	1236	0
27	15	65	0	0	1236	0
28	45	85	0	0	1236	0

Table A.4: *Benchmark MD_C104*

MD_C104				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	1236	0
1	45	68	10	0	1127	90
2	45	70	30	0	1125	90
3	42	66	10	0	1129	90
4	42	68	10	727	782	90
5	42	65	10	0	1130	90
6	40	69	20	0	1127	90
7	40	66	20	0	1130	90
8	38	68	20	255	324	90

(Continued on next page)

Table A.4 – *(continued from previous page)*

MD_C104				$D = 3$		
id	x	y	q	a	b	p
9	38	70	10	534	605	90
10	35	66	10	0	1129	90
11	35	69	10	448	505	90
12	25	85	20	0	1107	90
13	22	75	30	30	92	90
14	22	85	10	0	1106	90
15	20	80	40	384	429	90
16	20	85	40	0	1105	90
17	18	75	20	0	1112	90
18	15	75	20	0	1110	90
19	15	80	10	0	1106	90
20	30	50	10	0	1136	90
21	30	52	20	0	1135	90
22	28	52	20	0	1133	90
23	28	55	10	732	777	90
24	25	50	10	0	1131	90
25	25	52	40	169	224	90
26	40	50	0	0	1236	0
27	15	65	0	0	1236	0
28	45	85	0	0	1236	0

Table A.5: *Benchmark MD_C105*

MD_C105				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	1236	0
1	45	68	10	885	994	90
2	45	70	30	802	893	90
3	42	66	10	25	186	90
4	42	68	10	699	810	90
5	42	65	10	15	120	90
6	40	69	20	580	743	90
7	40	66	20	142	253	90
8	38	68	20	220	359	90
9	38	70	10	499	640	90
10	35	66	10	331	436	90
11	35	69	10	420	533	90
12	25	85	20	617	756	90
13	22	75	30	30	155	90

(Continued on next page)

Table A.5 – *(continued from previous page)*

MD_C105				$D = 3$		
id	x	y	q	a	b	p
14	22	85	10	541	646	90
15	20	80	40	362	451	90
16	20	85	40	448	555	90
17	18	75	20	75	172	90
18	15	75	20	142	291	90
19	15	80	10	244	379	90
20	30	50	10	10	137	90
21	30	52	20	888	991	90
22	28	52	20	776	919	90
23	28	55	10	709	800	90
24	25	50	10	25	184	90
25	25	52	40	142	251	90
26	40	50	0	0	1236	0
27	15	65	0	0	1236	0
28	45	85	0	0	1236	0

Table A.6: *Benchmark MD_C106*

MD_C106				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	1236	0
1	45	68	10	890	989	90
2	45	70	30	816	879	90
3	42	66	10	55	156	90
4	42	68	10	703	806	90
5	42	65	10	15	60	90
6	40	69	20	559	764	90
7	40	66	20	172	223	90
8	38	68	20	250	329	90
9	38	70	10	489	650	90
10	35	66	10	361	406	90
11	35	69	10	450	503	90
12	25	85	20	647	726	90
13	22	75	30	30	95	90
14	22	85	10	571	616	90
15	20	80	40	392	421	90
16	20	85	40	478	525	90
17	18	75	20	105	142	90
18	15	75	20	172	261	90

(Continued on next page)

Table A.6 – *(continued from previous page)*

MD_C106				$D = 3$		
id	x	y	q	a	b	p
19	15	80	10	274	349	90
20	30	50	10	10	77	90
21	30	52	20	918	961	90
22	28	52	20	806	889	90
23	28	55	10	739	770	90
24	25	50	10	55	154	90
25	25	52	40	172	221	90
26	40	50	0	0	1236	0
27	15	65	0	0	1236	0
28	45	85	0	0	1236	0

Table A.7: *Benchmark MD_C107*

MD_C107				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	1236	0
1	45	68	10	850	1030	90
2	45	70	30	758	938	90
3	42	66	10	16	196	90
4	42	68	10	665	845	90
5	42	65	10	15	195	90
6	40	69	20	572	752	90
7	40	66	20	108	288	90
8	38	68	20	200	380	90
9	38	70	10	480	660	90
10	35	66	10	294	474	90
11	35	69	10	387	567	90
12	25	85	20	597	777	90
13	22	75	30	30	210	90
14	22	85	10	504	684	90
15	20	80	40	317	497	90
16	20	85	40	412	592	90
17	18	75	20	34	214	90
18	15	75	20	127	307	90
19	15	80	10	222	402	90
20	30	50	10	10	190	90
21	30	52	20	850	1030	90
22	28	52	20	758	938	90
23	28	55	10	665	845	90

(Continued on next page)

Table A.7 – *(continued from previous page)*

MD_C107				$D = 3$		
id	x	y	q	a	b	p
24	25	50	10	15	195	90
25	25	52	40	107	287	90
26	40	50	0	0	1236	0
27	15	65	0	0	1236	0
28	45	85	0	0	1236	0

Table A.8: *Benchmark MD_C108*

MD_C108				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	1236	0
1	45	68	10	830	1049	90
2	45	70	30	756	939	90
3	42	66	10	16	336	90
4	42	68	10	643	866	90
5	42	65	10	15	226	90
6	40	69	20	499	824	90
7	40	66	20	87	308	90
8	38	68	20	150	429	90
9	38	70	10	429	710	90
10	35	66	10	279	488	90
11	35	69	10	363	590	90
12	25	85	20	547	826	90
13	22	75	30	30	280	90
14	22	85	10	489	698	90
15	20	80	40	318	495	90
16	20	85	40	394	609	90
17	18	75	20	33	226	90
18	15	75	20	68	365	90
19	15	80	10	176	447	90
20	30	50	10	10	265	90
21	30	52	20	836	1043	90
22	28	52	20	704	991	90
23	28	55	10	664	845	90
24	25	50	10	15	333	90
25	25	52	40	88	305	90
26	40	50	0	0	1236	0
27	15	65	0	0	1236	0
28	45	85	0	0	1236	0

Table A.9: *Benchmark MD_C109*

MD_C109				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	1236	0
1	45	68	10	760	1120	90
2	45	70	30	668	1028	90
3	42	66	10	16	376	90
4	42	68	10	575	935	90
5	42	65	10	15	375	90
6	40	69	20	482	842	90
7	40	66	20	18	378	90
8	38	68	20	110	470	90
9	38	70	10	390	750	90
10	35	66	10	204	564	90
11	35	69	10	297	657	90
12	25	85	20	507	867	90
13	22	75	30	30	390	90
14	22	85	10	414	774	90
15	20	80	40	227	587	90
16	20	85	40	322	682	90
17	18	75	20	33	393	90
18	15	75	20	37	397	90
19	15	80	10	132	492	90
20	30	50	10	10	370	90
21	30	52	20	760	1120	90
22	28	52	20	668	1028	90
23	28	55	10	575	935	90
24	25	50	10	15	375	90
25	25	52	40	17	377	90
26	40	50	0	0	1236	0
27	15	65	0	0	1236	0
28	45	85	0	0	1236	0

A.2 Benchmarks MD_R

Table A.10: *Benchmark MD_R101*

MD_R101				$D = 8$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	161	171	10

(Continued on next page)

Table A.10 – *(continued from previous page)*

MD_R101				$D = 8$		
id	x	y	q	a	b	p
2	35	17	7	50	60	10
3	55	45	13	116	126	10
4	55	20	19	149	159	10
5	15	30	26	34	44	10
6	25	30	3	99	109	10
7	20	50	5	81	91	10
8	10	43	9	95	105	10
9	55	60	16	97	107	10
10	30	60	16	124	134	10
11	20	65	12	67	77	10
12	50	35	19	63	73	10
13	30	25	23	159	169	10
14	15	10	20	32	42	10
15	30	5	8	61	71	10
16	10	20	19	75	85	10
17	5	30	2	157	167	10
18	20	40	12	87	97	10
19	15	60	17	76	86	10
20	45	65	9	126	136	10
21	45	20	11	62	72	10
22	45	10	18	97	107	10
23	55	5	29	68	78	10
24	65	35	3	153	163	10
25	65	20	6	172	182	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0
30	40	20	0	0	230	0
31	45	50	0	0	230	0
32	25	50	0	0	230	0
33	25	20	0	0	230	0

Table A.11: *Benchmark MD_R102*

MD_R102				$D = 8$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	0	204	10

(Continued on next page)

Table A.11 – *(continued from previous page)*

MD_R102				$D = 8$		
id	x	y	q	a	b	p
2	35	17	7	0	202	10
3	55	45	13	0	197	10
4	55	20	19	149	159	10
5	15	30	26	0	199	10
6	25	30	3	99	109	10
7	20	50	5	0	198	10
8	10	43	9	95	105	10
9	55	60	16	97	107	10
10	30	60	16	124	134	10
11	20	65	12	67	77	10
12	50	35	19	0	205	10
13	30	25	23	159	169	10
14	15	10	20	32	42	10
15	30	5	8	61	71	10
16	10	20	19	75	85	10
17	5	30	2	157	167	10
18	20	40	12	87	97	10
19	15	60	17	76	86	10
20	45	65	9	126	136	10
21	45	20	11	0	201	10
22	45	10	18	97	107	10
23	55	5	29	68	78	10
24	65	35	3	153	163	10
25	65	20	6	172	182	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0
30	40	20	0	0	230	0
31	45	50	0	0	230	0
32	25	50	0	0	230	0
33	25	20	0	0	230	0

Table A.12: *Benchmark MD_R103*

MD_R103				$D = 5$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	0	204	10

(Continued on next page)

Table A.12 – *(continued from previous page)*

MD_R103				$D = 5$		
id	x	y	q	a	b	p
2	35	17	7	0	202	10
3	55	45	13	0	197	10
4	55	20	19	149	159	10
5	15	30	26	0	199	10
6	25	30	3	99	109	10
7	20	50	5	0	198	10
8	10	43	9	95	105	10
9	55	60	16	97	107	10
10	30	60	16	124	134	10
11	20	65	12	67	77	10
12	50	35	19	0	205	10
13	30	25	23	159	169	10
14	15	10	20	0	187	10
15	30	5	8	61	71	10
16	10	20	19	0	190	10
17	5	30	2	157	167	10
18	20	40	12	0	204	10
19	15	60	17	0	187	10
20	45	65	9	0	188	10
21	45	20	11	0	201	10
22	45	10	18	97	107	10
23	55	5	29	68	78	10
24	65	35	3	0	190	10
25	65	20	6	172	182	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0
30	35	35	0	0	230	0

Table A.13: *Benchmark MD_R104*

MD_R104				$D = 4$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	0	204	10
2	35	17	7	0	202	10
3	55	45	13	0	197	10
4	55	20	19	149	159	10

(Continued on next page)

Table A.13 – *(continued from previous page)*

MD_R104				$D = 4$		
id	x	y	q	a	b	p
5	15	30	26	0	199	10
6	25	30	3	0	208	10
7	20	50	5	0	198	10
8	10	43	9	95	105	10
9	55	60	16	97	107	10
10	30	60	16	0	194	10
11	20	65	12	67	77	10
12	50	35	19	0	205	10
13	30	25	23	159	169	10
14	15	10	20	0	187	10
15	30	5	8	61	71	10
16	10	20	19	0	190	10
17	5	30	2	0	189	10
18	20	40	12	0	204	10
19	15	60	17	0	187	10
20	45	65	9	0	188	10
21	45	20	11	0	201	10
22	45	10	18	0	193	10
23	55	5	29	68	78	10
24	65	35	3	0	190	10
25	65	20	6	172	182	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0

Table A.14: *Benchmark MD_R105*

MD_R105				$D = 6$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	151	181	10
2	35	17	7	40	70	10
3	55	45	13	106	136	10
4	55	20	19	139	169	10
5	15	30	26	24	54	10
6	25	30	3	89	119	10
7	20	50	5	71	101	10
8	10	43	9	85	115	10

(Continued on next page)

Table A.14 – *(continued from previous page)*

MD_R105				$D = 6$		
id	x	y	q	a	b	p
9	55	60	16	87	117	10
10	30	60	16	114	144	10
11	20	65	12	57	87	10
12	50	35	19	53	83	10
13	30	25	23	149	179	10
14	15	10	20	32	62	10
15	30	5	8	51	81	10
16	10	20	19	65	95	10
17	5	30	2	147	177	10
18	20	40	12	77	107	10
19	15	60	17	66	96	10
20	45	65	9	116	146	10
21	45	20	11	52	82	10
22	45	10	18	87	117	10
23	55	5	29	58	88	10
24	65	35	3	143	173	10
25	65	20	6	156	186	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0
30	30	35	0	0	230	0
31	40	35	0	0	230	0

Table A.15: *Benchmark MD_R106*

MD_R106				$D = 5$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	0	204	10
2	35	17	7	0	202	10
3	55	45	13	0	197	10
4	55	20	19	139	169	10
5	15	30	26	0	199	10
6	25	30	3	89	119	10
7	20	50	5	0	198	10
8	10	43	9	85	115	10
9	55	60	16	87	117	10
10	30	60	16	114	144	10

(Continued on next page)

Table A.15 – *(continued from previous page)*

MD_R106				$D = 5$		
id	x	y	q	a	b	p
11	20	65	12	57	87	10
12	50	35	19	0	205	10
13	30	25	23	149	179	10
14	15	10	20	32	62	10
15	30	5	8	51	81	10
16	10	20	19	65	95	10
17	5	30	2	147	177	10
18	20	40	12	77	107	10
19	15	60	17	66	96	10
20	45	65	9	116	146	10
21	45	20	11	0	201	10
22	45	10	18	87	117	10
23	55	5	29	58	88	10
24	65	35	3	143	173	10
25	65	20	6	156	186	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0
30	35	35	0	0	230	0

Table A.16: *Benchmark MD_R107*

MD_R107				$D = 4$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	0	204	10
2	35	17	7	0	202	10
3	55	45	13	0	197	10
4	55	20	19	139	169	10
5	15	30	26	0	199	10
6	25	30	3	89	119	10
7	20	50	5	0	198	10
8	10	43	9	85	115	10
9	55	60	16	87	117	10
10	30	60	16	114	144	10
11	20	65	12	57	87	10
12	50	35	19	0	205	10
13	30	25	23	149	179	10

(Continued on next page)

Table A.16 – *(continued from previous page)*

MD_R107				$D = 4$		
id	x	y	q	a	b	p
14	15	10	20	0	187	10
15	30	5	8	51	81	10
16	10	20	19	0	190	10
17	5	30	2	147	177	10
18	20	40	12	0	204	10
19	15	60	17	0	187	10
20	45	65	9	0	188	10
21	45	20	11	0	201	10
22	45	10	18	87	117	10
23	55	5	29	58	88	10
24	65	35	3	0	190	10
25	65	20	6	156	186	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0

Table A.17: *Benchmark MD_R108*

MD_R108				$D = 4$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	0	204	10
2	35	17	7	0	202	10
3	55	45	13	0	197	10
4	55	20	19	139	169	10
5	15	30	26	0	199	10
6	25	30	3	0	208	10
7	20	50	5	0	198	10
8	10	43	9	85	115	10
9	55	60	16	87	117	10
10	30	60	16	0	194	10
11	20	65	12	57	87	10
12	50	35	19	0	205	10
13	30	25	23	149	179	10
14	15	10	20	0	187	10
15	30	5	8	51	81	10
16	10	20	19	0	190	10
17	5	30	2	0	189	10

(Continued on next page)

Table A.17 – *(continued from previous page)*

MD_R108				$D = 4$		
id	x	y	q	a	b	p
18	20	40	12	0	204	10
19	15	60	17	0	187	10
20	45	65	9	0	188	10
21	45	20	11	0	201	10
22	45	10	18	0	193	10
23	55	5	29	58	88	10
24	65	35	3	0	190	10
25	65	20	6	156	186	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0

Table A.18: *Benchmark MD_R109*

MD_R109				$D = 5$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	133	198	10
2	35	17	7	22	87	10
3	55	45	13	98	143	10
4	55	20	19	123	184	10
5	15	30	26	20	93	10
6	25	30	3	76	131	10
7	20	50	5	61	110	10
8	10	43	9	75	124	10
9	55	60	16	74	129	10
10	30	60	16	107	150	10
11	20	65	12	42	101	10
12	50	35	19	38	97	10
13	30	25	23	131	196	10
14	15	10	20	32	114	10
15	30	5	8	35	96	10
16	10	20	19	52	107	10
17	5	30	2	124	189	10
18	20	40	12	69	114	10
19	15	60	17	52	109	10
20	45	65	9	105	156	10
21	45	20	11	37	96	10

(Continued on next page)

Table A.18 – *(continued from previous page)*

MD_R109				$D = 5$		
id	x	y	q	a	b	p
22	45	10	18	76	127	10
23	55	5	29	43	102	10
24	65	35	3	124	190	10
25	65	20	6	121	186	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0
30	35	35	0	0	230	0

Table A.19: *Benchmark MD_R110*

MD_R110				$D = 5$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	130	201	10
2	35	17	7	20	89	10
3	55	45	13	106	135	10
4	55	20	19	71	195	10
5	15	30	26	20	107	10
6	25	30	3	54	153	10
7	20	50	5	66	105	10
8	10	43	9	61	138	10
9	55	60	16	53	150	10
10	30	60	16	101	156	10
11	20	65	12	33	152	10
12	50	35	19	38	97	10
13	30	25	23	70	208	10
14	15	10	20	32	137	10
15	30	5	8	30	154	10
16	10	20	19	54	105	10
17	5	30	2	51	189	10
18	20	40	12	77	106	10
19	15	60	17	53	108	10
20	45	65	9	109	152	10
21	45	20	11	37	96	10
22	45	10	18	59	144	10
23	55	5	29	36	155	10
24	65	35	3	118	190	10

(Continued on next page)

Table A.19 – *(continued from previous page)*

MD_R110				$D = 5$		
id	x	y	q	a	b	p
25	65	20	6	47	186	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0
30	35	35	0	0	230	0

Table A.20: *Benchmark MD_R111*

MD_R111				$D = 5$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	15	204	10
2	35	17	7	18	202	10
3	55	45	13	54	187	10
4	55	20	19	138	169	10
5	15	30	26	20	199	10
6	25	30	3	76	131	10
7	20	50	5	21	170	10
8	10	43	9	87	112	10
9	55	60	16	88	115	10
10	30	60	16	107	150	10
11	20	65	12	57	86	10
12	50	35	19	15	192	10
13	30	25	23	147	180	10
14	15	10	20	32	187	10
15	30	5	8	50	81	10
16	10	20	19	29	139	10
17	5	30	2	124	189	10
18	20	40	12	47	136	10
19	15	60	17	32	146	10
20	45	65	9	79	182	10
21	45	20	11	18	195	10
22	45	10	18	76	127	10
23	55	5	29	58	87	10
24	65	35	3	58	190	10
25	65	20	6	153	186	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0

(Continued on next page)

Table A.20 – *(continued from previous page)*

MD_R111				$D = 5$		
id	x	y	q	a	b	p
28	10	60	0	0	230	0
29	10	10	0	0	230	0
30	35	35	0	0	230	0

Table A.21: *Benchmark MD_R112*

MD_R112				$D = 4$		
id	x	y	q	a	b	p
0	35	35	0	0	230	0
1	41	49	10	73	204	10
2	35	17	7	18	147	10
3	55	45	13	76	165	10
4	55	20	19	73	195	10
5	15	30	26	20	167	10
6	25	30	3	49	158	10
7	20	50	5	36	135	10
8	10	43	9	50	149	10
9	55	60	16	47	156	10
10	30	60	16	85	172	10
11	20	65	12	33	152	10
12	50	35	19	15	133	10
13	30	25	23	79	208	10
14	15	10	20	32	187	10
15	30	5	8	30	152	10
16	10	20	19	29	139	10
17	5	30	2	60	189	10
18	20	40	12	47	136	10
19	15	60	17	32	146	10
20	45	65	9	79	182	10
21	45	20	11	18	136	10
22	45	10	18	50	153	10
23	55	5	29	36	155	10
24	65	35	3	58	190	10
25	65	20	6	56	186	10
26	60	10	0	0	230	0
27	60	60	0	0	230	0
28	10	60	0	0	230	0
29	10	10	0	0	230	0

A.3 Benchmarks MD_RC

Table A.22: *Benchmark MD_RC101*

MD_RC101				$D = 5$		
id	x	y	q	a	b	p
0	40	50	0	0	240	0
1	25	85	20	145	175	10
2	22	75	30	50	80	10
3	22	85	10	109	139	10
4	20	80	40	141	171	10
5	20	85	20	41	71	10
6	18	75	20	95	125	10
7	15	75	20	79	109	10
8	15	80	10	91	121	10
9	10	35	20	91	121	10
10	10	40	30	119	149	10
11	8	40	40	59	89	10
12	8	45	20	64	94	10
13	5	35	10	142	172	10
14	5	45	10	35	65	10
15	2	40	20	58	88	10
16	0	40	20	72	102	10
17	0	45	20	149	179	10
18	44	5	20	87	117	10
19	42	10	40	72	102	10
20	42	15	10	122	152	10
21	40	5	10	67	97	10
22	40	15	40	92	122	10
23	38	5	30	65	95	10
24	38	15	10	148	178	10
25	35	5	20	154	184	10
26	40	50	0	0	240	0
27	5	75	0	0	240	0
28	5	15	0	0	240	0
29	20	60	0	0	240	0
30	20	20	0	0	240	0

Table A.23: *Benchmark MD_RC102*

MD_RC102				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	240	0
1	25	85	20	0	191	10
2	22	75	30	0	199	10
3	22	85	10	0	190	10
4	20	80	40	141	171	10
5	20	85	20	0	189	10
6	18	75	20	95	125	10
7	15	75	20	0	194	10
8	15	80	10	91	121	10
9	10	35	20	91	121	10
10	10	40	30	119	149	10
11	8	40	40	59	89	10
12	8	45	20	0	197	10
13	5	35	10	142	172	10
14	5	45	10	35	65	10
15	2	40	20	58	88	10
16	0	40	20	72	102	10
17	0	45	20	149	179	10
18	44	5	20	87	117	10
19	42	10	40	72	102	10
20	42	15	10	122	152	10
21	40	5	10	0	185	10
22	40	15	40	92	122	10
23	38	5	30	65	95	10
24	38	15	10	148	178	10
25	35	5	20	154	184	10
26	40	50	0	0	240	0
27	5	75	0	0	240	0
28	5	15	0	0	240	0

Table A.24: *Benchmark MD_RC103*

MD_RC103				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	240	0
1	25	85	20	0	191	10
2	22	75	30	0	199	10
3	22	85	10	0	190	10

(Continued on next page)

Table A.24 – *(continued from previous page)*

MD_RC103				$D = 3$		
id	x	y	q	a	b	p
4	20	80	40	141	171	10
5	20	85	20	0	189	10
6	18	75	20	95	125	10
7	15	75	20	0	194	10
8	15	80	10	91	121	10
9	10	35	20	91	121	10
10	10	40	30	119	149	10
11	8	40	40	59	89	10
12	8	45	20	0	197	10
13	5	35	10	142	172	10
14	5	45	10	0	194	10
15	2	40	20	58	88	10
16	0	40	20	0	188	10
17	0	45	20	149	179	10
18	44	5	20	0	184	10
19	42	10	40	0	189	10
20	42	15	10	0	194	10
21	40	5	10	0	185	10
22	40	15	40	92	122	10
23	38	5	30	65	95	10
24	38	15	10	0	194	10
25	35	5	20	154	184	10
26	40	50	0	0	240	0
27	5	75	0	0	240	0
28	5	15	0	0	240	0

Table A.25: *Benchmark MD_RC104*

MD_RC104				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	240	0
1	25	85	20	0	191	10
2	22	75	30	0	199	10
3	22	85	10	0	190	10
4	20	80	40	141	171	10
5	20	85	20	0	189	10
6	18	75	20	0	196	10
7	15	75	20	0	194	10
8	15	80	10	91	121	10

(Continued on next page)

Table A.25 – *(continued from previous page)*

MD_RC104				$D = 3$		
id	x	y	q	a	b	p
9	10	35	20	91	121	10
10	10	40	30	0	198	10
11	8	40	40	59	89	10
12	8	45	20	0	197	10
13	5	35	10	142	172	10
14	5	45	10	0	194	10
15	2	40	20	58	88	10
16	0	40	20	0	188	10
17	0	45	20	0	189	10
18	44	5	20	0	184	10
19	42	10	40	0	189	10
20	42	15	10	0	194	10
21	40	5	10	0	185	10
22	40	15	40	0	195	10
23	38	5	30	65	95	10
24	38	15	10	0	194	10
25	35	5	20	154	184	10
26	40	50	0	0	240	0
27	5	75	0	0	240	0
28	5	15	0	0	240	0

Table A.26: *Benchmark MD_RC105*

MD_RC105				$D = 4$		
id	x	y	q	a	b	p
0	40	50	0	0	240	0
1	25	85	20	71	191	10
2	22	75	30	30	150	10
3	22	85	10	64	184	10
4	20	80	40	151	161	10
5	20	85	20	40	160	10
6	18	75	20	96	123	10
7	15	75	20	35	155	10
8	15	80	10	101	111	10
9	10	35	20	101	111	10
10	10	40	30	123	144	10
11	8	40	40	69	79	10
12	8	45	20	32	152	10
13	5	35	10	152	162	10

(Continued on next page)

Table A.26 – *(continued from previous page)*

MD_RC105				$D = 4$		
id	x	y	q	a	b	p
14	5	45	10	35	117	10
15	2	40	20	68	78	10
16	0	40	20	59	114	10
17	0	45	20	147	180	10
18	44	5	20	79	124	10
19	42	10	40	58	115	10
20	42	15	10	111	162	10
21	40	5	10	45	165	10
22	40	15	40	94	119	10
23	38	5	30	75	85	10
24	38	15	10	128	194	10
25	35	5	20	171	181	10
26	40	50	0	0	240	0
27	5	75	0	0	240	0
28	5	15	0	0	240	0
29	20	50	0	0	240	0

Table A.27: *Benchmark MD_RC106*

MD_RC106				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	240	0
1	25	85	20	130	190	10
2	22	75	30	35	95	10
3	22	85	10	94	154	10
4	20	80	40	126	186	10
5	20	85	20	40	100	10
6	18	75	20	80	140	10
7	15	75	20	64	124	10
8	15	80	10	76	136	10
9	10	35	20	76	136	10
10	10	40	30	104	164	10
11	8	40	40	44	104	10
12	8	45	20	49	109	10
13	5	35	10	127	187	10
14	5	45	10	35	95	10
15	2	40	20	43	103	10
16	0	40	20	57	117	10
17	0	45	20	129	189	10

(Continued on next page)

Table A.27 – *(continued from previous page)*

MD_RC106				$D = 3$		
id	x	y	q	a	b	p
18	44	5	20	72	132	10
19	42	10	40	57	117	10
20	42	15	10	107	167	10
21	40	5	10	52	112	10
22	40	15	40	77	137	10
23	38	5	30	50	110	10
24	38	15	10	133	193	10
25	35	5	20	124	184	10
26	40	50	0	0	240	0
27	5	75	0	0	240	0
28	5	15	0	0	240	0

Table A.28: *Benchmark MD_RC107*

MD_RC107				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	240	0
1	25	85	20	125	191	10
2	22	75	30	32	97	10
3	22	85	10	101	146	10
4	20	80	40	71	193	10
5	20	85	20	40	113	10
6	18	75	20	55	164	10
7	15	75	20	69	118	10
8	15	80	10	56	155	10
9	10	35	20	51	160	10
10	10	40	30	90	177	10
11	8	40	40	33	152	10
12	8	45	20	49	108	10
13	5	35	10	62	191	10
14	5	45	10	35	117	10
15	2	40	20	39	161	10
16	0	40	20	59	114	10
17	0	45	20	60	189	10
18	44	5	20	79	124	10
19	42	10	40	58	115	10
20	42	15	10	111	162	10
21	40	5	10	52	111	10
22	40	15	40	55	158	10

(Continued on next page)

Table A.28 – *(continued from previous page)*

MD_RC107				$D = 3$		
id	x	y	q	a	b	p
23	38	5	30	45	164	10
24	38	15	10	128	194	10
25	35	5	20	54	184	10
26	40	50	0	0	240	0
27	5	75	0	0	240	0
28	5	15	0	0	240	0

Table A.29: *Benchmark MD_RC108*

MD_RC108				$D = 3$		
id	x	y	q	a	b	p
0	40	50	0	0	240	0
1	25	85	20	49	191	10
2	22	75	30	30	168	10
3	22	85	10	95	152	10
4	20	80	40	69	193	10
5	20	85	20	40	189	10
6	18	75	20	60	159	10
7	15	75	20	54	133	10
8	15	80	10	67	144	10
9	10	35	20	57	154	10
10	10	40	30	106	161	10
11	8	40	40	33	152	10
12	8	45	20	32	148	10
13	5	35	10	53	191	10
14	5	45	10	35	194	10
15	2	40	20	39	163	10
16	0	40	20	41	141	10
17	0	45	20	51	189	10
18	44	5	20	73	130	10
19	42	10	40	40	148	10
20	42	15	10	94	179	10
21	40	5	10	45	161	10
22	40	15	40	64	149	10
23	38	5	30	45	164	10
24	38	15	10	51	194	10
25	35	5	20	45	183	10
26	40	50	0	0	240	0
27	5	75	0	0	240	0
28	5	15	0	0	240	0

Appendix B

Acronyms

Table B.1: *Acronyms*

Acronym	Description
ACO	Ant Colony Optimization
AMP	Adaptive Memory Programming
DARP	Dial-a-Ride Problem
DBGP	Dynamic Benchmark Generator for Permutation
DC	Data Component
DM	Decision Maker
DSS	Decision Support System
DVRP	Dynamic Vehicle Routing Problem
EV	Electric Vehicle
GA	Genetic Algorithm
GPS	Global Positioning System
GVRP	Green Vehicle Routing Problem
HU	Handling Unit
IC	Interface Component
ICT	Information Communication Technologies

(Continued on next page)

Table B.1 – *(continued from previous page)*

Acronym	Description
ILP	Integer Linear Programming
ITS	Intelligent Transport System
MAS	Multi-Agent System
MC	Model Component
MDVRPTW	Multi-depot Vehicle Routing Problem with Time Windows
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Programming
OEM	Original Equipment Manufacturer
OV	Opportunity Valuation
PAC	Post Automation Center
PD	Postal Delivery
PDC	Postal Delivery Center
PRP	Pollution Routing Problem
PSO	Particle Swarm Optimization
SD-VNS	Stochastic Dynamic Variable Neighborhood Search
S-VNS	Stochastic Variable Neighborhood Search
TRP	Traveling Repairman Problem
TS	Tabu Search
TSP	Traveling Salesman Problem
TSPSTW	Traveling Salesman Problem with soft Time Windows
TSPTW	Traveling Salesman Problem with Time Windows
UML	Unified Modeling Language
VNS	Variable Neighborhood Search
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
WC	Waste Collection

(Continued on next page)

Table B.1 – *(continued from previous page)*

Acronym	Description
WCRP	Waste Collection Routing Problem

Bibliography

[1] R. Lahyani, M. Khemakhem, and F. Semet, "Rich vehicle routing problems: From a taxonomy to a definition," *European Journal of Operational Research*, vol. 241, no. 1, pp. 1 – 14, 2015.

[2] C. Lin, K. Choy, G. Ho, S. Chung, and H. Lam, "Survey of green vehicle routing problem: Past and future trends," *Expert Systems with Applications*, vol. 41, no. 4, Part 1, pp. 1118 – 1138, 2014.

[3] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1 – 11, 2013.

[4] H. Psaraftis, "Dynamic vehicle routing: Status and prospects," *Annals of Operations Research*, vol. 61, pp. 143 – 164, 1995.

[5] S. Ichoua, M. Gendreau, and J.-Y. Potvin, *Planned Route Optimization For Real-Time Vehicle Routing*. Boston, MA: Springer US, 2007, pp. 1–18. [Online]. Available: https://doi.org/10.1007/978-0-387-71722-7_1

[6] H. N. Psaraftis, M. Wen, and C. A. Kontovas, "Dynamic vehicle routing problems: Three decades and counting," *Networks*, vol. 67, no. 1, pp. 3–31, 2016. [Online]. Available: http://dx.doi.org/10.1002/net.21628

[7] M. Fanti, M. Franceschelli, A. Mangini, G. Pedroncelli, and W. Ukovich, "Discrete consensus in networks with constrained capacity," in *Proceedings of the IEEE Conference on Decision and Control*, 2013, pp. 2012–2017.

[8] M. P. Fanti, A. M. Mangini, G. Pedroncelli, and W. Ukovich, "Discrete consensus for asynchronous distributed task assignment," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 251–255.

[9] J. A. Orozco and J. Barceló, "Reactive and proactive routing strategies with real-time traffic information," *Procedia - Social and Behavioral Sciences*, vol. 39, pp. 633 – 648, 2012, seventh International Conference on City Logistics which was held on June 7- 9,2011, Mallorca, Spain.

[10] F. Ferrucci and S. Bock, "A general approach for controlling vehicle en-route diversions in dynamic vehicle routing problems," *Transportation Research Part B: Methodological*, vol. 77, pp. 76 – 87, 2015.

[11] H. Grzybowska and J. Barceló, "Decision support system for real-time urban freight management," *Procedia - Social and Behavioral Sciences*, vol. 39, pp. 712 – 725, 2012, seventh International Conference on City Logistics which was held on June 7-9,2011, Mallorca, Spain.

[12] N. Wilson, N. Colvin, M. I. of Technology. Center for Transportation Studies, U. S. U. M. T. Administration, and R. G. R. T. Authority, *Computer control of the Rochester dial-a-ride system*. Massachusetts Institute of Technology, Center for Transportation Studies, 1977.

[13] H. Psaraftis, "A dynamic-programming solution to the single vehicle many-to-many immediate request dial-a-ride problem," *Transportation Science*, vol. 14, no. 2, pp. 130 – 154, 1980.

[14] M. Barkaoui and M. Gendreau, "An adaptive evolutionary approach for real-time vehicle routing and dispatching," *Computers & Operations Research*, vol. 40, no. 7, pp. 1766 – 1776, 2013.

[15] F. Ferrucci and S. Bock, "Real-time control of express pickup and delivery processes in a dynamic environment," *Transportation Research Part B: Methodological*, vol. 63, pp. 1 – 14, 2014.

[16] X. Hu, L. Sun, and L. Liu, "A PAM approach to handling disruptions in real-time vehicle routing problems," *Decision Support Systems*, vol. 54, no. 3, pp. 1380 – 1393, 2013.

[17] M. Mavrovouniotis and S. Yang, "Ant algorithms with immigrants schemes for the dynamic vehicle routing problem," *Information Sciences*, vol. 294, pp. 456 – 477, 2015, innovative Applications of Artificial Neural Networks in Engineering.

[18] B. Sarasola, K. F. Doerner, V. Schmid, and E. Alba, "Variable neighborhood search for the stochastic and dynamic vehicle routing problem," *Annals of Operations Research*, no. 236, pp. 425 – 461, 2016.

[19] C. Lin, K. Choy, G. Ho, H. Lam, G. K. Pang, and K. Chin, "A decision support system for optimizing dynamic courier routing operations," *Expert Systems with Applications*, vol. 41, no. 15, pp. 6917 – 6933, 2014.

[20] A. N. G. Novaes, E. T. Bez, and P. J. Burin, *Dynamics in Logistics: Third International Conference, LDIC 2012 Bremen, Germany, February/March 2012 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ch. Fault Detection in Dynamic Vehicle Routing Operations, pp. 13–32.

[21] D. P. Aragão, A. N. G. Novaes, and M. M. Mendes Luna, "A multi agent based system to enable dynamic vehicle routing," *TRANSPORTES*, vol. 23, no. 1, pp. 69 – 77, 2015.

[22] S. Binart, P. Dejax, M. Gendreau, and F. Semet, "A 2-stage method for a field service routing problem with stochastic travel and service times," *Computers & Operations Research*, vol. 65, pp. 64 – 75, 2016.

[23] S. Ghannadpour, S. Noori, and R. Tavakkoli-Moghaddam, "Multiobjective dynamic vehicle routing problem with fuzzy travel times and customers. satisfaction in supply chain management," *Engineering Management, IEEE Transactions on*, vol. 60, no. 4, pp. 777–790, Nov 2013.

[24] S. F. Ghannadpour, S. Noori, R. Tavakkoli-Moghaddam, and K. Ghoseiri, "A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application," *Applied Soft Computing*, vol. 14, Part C, pp. 504 – 527, 2014.

[25] J. de Armas and B. Melián-Batista, "Variable neighborhood search for a dynamic rich vehicle routing problem with time windows," *Computers & Industrial Engineering*, vol. 85, pp. 120 – 131, 2015.

[26] ——, "Constrained dynamic vehicle routing problems with time windows," *Soft Computing*, vol. 19, pp. 2481 – 2498, 2015.

[27] M. Albareda-Sambola, E. Fernández, and G. Laporte, "The dynamic multiperiod vehicle routing problem with probabilistic information," *Computers & Operations Research*, vol. 48, pp. 31 – 39, 2014.

[28] M. Okulewicz and J. Mańdziuk, *Artificial Intelligence and Soft Computing: 12th International Conference, ICAISC 2013, Zakopane, Poland, June 9-13, 2013, Proceedings, Part II.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ch. Application of Particle Swarm Optimization Algorithm to Dynamic Vehicle Routing Problem, pp. 547–558.

[29] ——, "Two-phase multi-swarm pso and the dynamic vehicle routing problem," in *Computational Intelligence for Human-like Intelligence (CIHLI), 2014 IEEE Symposium on*, Dec 2014, pp. 1–8.

[30] M. Schyns, "An ant colony system for responsive dynamic vehicle routing," *European Journal of Operational Research*, vol. 245, no. 3, pp. 704 – 718, 2015.

[31] J. Euchi, A. Yassine, and H. Chabchoub, "The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach," *Swarm and Evolutionary Computation*, vol. 21, pp. 41 – 53, 2015.

[32] M. Gath, S. Edelkamp, and O. Herzog, "Agent-based dispatching in groupage traffic," in *Computational Intelligence In Production And Logistics Systems (CIPLS), 2013 IEEE Workshop on*, April 2013, pp. 54–60.

[33] Q. Xiongwen and X. Ya, "Dynamic pick-up and delivery vehicle routing problem with ready-time and deadline," in *Control Conference (CCC), 2013 32nd Chinese*, July 2013, pp. 2515–2520.

[34] T. Warden and J. Wojtusiak, "Learnable evolutionary optimization in autonomous pickup & delivery planning: a scenario, system architecture and initial results," *TZI-Bericht Nr*, vol. 55, 2010.

[35] F. Ferrucci, S. Bock, and M. Gendreau, "A pro-active real-time control approach for dynamic vehicle routing problems dealing with the delivery of urgent goods," *European Journal of Operational Research*, vol. 225, no. 1, pp. 130 – 141, 2013.

[36] M. Barkaoui, J. Berger, and A. Boukhtouta, "Customer satisfaction in dynamic vehicle routing problem with time windows," *Applied Soft Computing*, vol. 35, pp. 423 – 432, 2015.

[37] M. Bruni, F. Guerriero, and P. Beraldi, "Designing robust routes for demand-responsive transport systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 70, pp. 1 – 16, 2014.

[38] M. Mes, M. van der Heijden, and P. Schuur, "Interaction between intelligent agent strategies for real-time transportation planning," *Central European Journal of Operations Research*, vol. 21, pp. 337 – 358, 2013.

[39] D. Muñoz-Carpintero, D. Sáez, C. E. Cortés, and A. Núñez, "A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach," *Transportation Science*, vol. 49, no. 2, pp. 239–253, 2015.

[40] S. Nambiar and S. Idicula, "A multi-agent vehicle routing system for garbage collection," in *Advanced Computing (ICoAC), 2013 Fifth International Conference on*, Dec 2013, pp. 72–76.

[41] G. Kim, Y.-S. Ong, C. K. Heng, P. S. Tan, and N. Zhang, "City vehicle routing problem (city VRP): A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1654–1666, 2015.

[42] M. S. A. Larsen, O. Madsen, "Partially dynamic vehicle routing-models and algorithms," *The Journal of the Operational Research Society*, vol. 53, no. 6, pp. 637–646, 2002.

[43] L. Coslovich, R. Pesenti, and W. Ukovich, "A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem," *European Journal of Operational Research*, vol. 175, no. 3, pp. 1605 – 1615, 2006.

[44] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem," *Journal of Combinatorial Optimization*, vol. 10, no. 4, pp. 327–343, 2005.

[45] E. D. Taillard, L. M. Gambardella, M. Gendreau, and J.-Y. Potvin, "Adaptive memory programming: A unified view of metaheuristics," *European Journal of Operational Research*, vol. 135, no. 1, pp. 1–16, 2001.

[46] M. Gendreau, F. Guertin, J.-Y. Potvin, and E. Taillard, "Parallel tabu search for real-time vehicle routing and dispatching," *Transportation Science*, vol. 33, no. 4, pp. 381–390, 1999.

[47] R. W. Bent and P. V. Hentenryck, "Scenario-based planning for partially dynamic vehicle routing with stochastic customers," *Operations Research*, vol. 52, pp. 977–987, 2003.

[48] T. L. Lai, *Sequential Analysis*. John Wiley & Sons, Ltd, 2005.

[49] L. M. Ausubel and P. Milgrom, "The lovely but lonely vickrey auction," *Combinatorial Auctions*, vol. 17, 2006.

[50] G. Kim, Y.-S. Ong, C. K. Heng, P. S. Tan, and N. Zhang, "City vehicle routing problem (city vrp): A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1654–1666, 2015.

[51] I. von Poser and A. Awad, "Optimal routing for solid waste collection in cities by using real genetic algorithm," in *2006. ICTTA '06. 2nd Information and Communication Technologies*, vol. 1, 2006, pp. 221–226.

[52] A. Singh, G. Ruhe, S. Amereei, and S. Banack, "Decision support for capacitated arc routing for providing municipal waste and recycling services," in *2014 47th Hawaii International Conference on System Sciences (HICSS)*, 2014, pp. 986–993.

[53] J. Liu, D. Liu, M. Liu, and Y. He, "An improved multiple ant colony system for the collection vehicle routing problems with intermediate facilities," in *2010 8th World Congress on Intelligent Control and Automation (WCICA)*, 2010, pp. 3078–3083.

[54] J. Liu and Y. He, "A clustering-based multiple ant colony system for the waste collection vehicle routing problems," in *2012 Fifth International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2, 2012, pp. 182–185.

[55] J. Bautista, E. Fernández, and J. Pereira, "Solving an urban waste collection problem using ants heuristics," *Computers & Operations Research*, vol. 35, no. 9, pp. 3020 – 3033, 2008.

[56] R. Revetria, A. Testa, and L. Cassettari, "A generalized simulation framework to manage logistics systems: a case study in waste management and environmental protection," in *Proceedings of the 2011 Winter Simulation Conference (WSC)*, 2011, pp. 943–952.

[57] L. Abbatecola, M. P. Fanti, A. M. Mangini, and W. Ukovich, "A decision support approach for postal delivery and waste collection services," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1458–1470, Oct 2016.

[58] M. Junger, G. Reinelt, and G. Rinaldi, *The Traveling Salesman Problem.* North-Holland, Amsterdam: M.O. Ball, T.L. Magnanti, C.L. Monma, G.L.Nemhauser

(Eds.), Handbooks in Operations Research and Management Science, 1997, vol. vol. 7, Network Models.

[59] F. Sperandio, C. Gomes, J. Borges, A. Carvalho Brito, and B. Almada-Lobo, "An intelligent decision support system for the operating theater: A case study," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 265–273, 2014.

[60] V. Boschian, M. Dotoli, M. Fanti, G. Iacobellis, and W. Ukovich, "A metamodeling approach to the management of intermodal transportation networks," *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 3, pp. 457–469, 2011.

[61] R. Miles and K. Hamilton, *Learning UML 2.0.* Sabastopol, CA: OReilly Media, 2006.

[62] S. N. Parragh, k.F. Doerner, and R. Hartl, "A survey on pickup and delivery problems. part ii: Transportation between pickup and delivery locations," *Journal für Betriebswirtschaft*, vol. 58, pp. 21–51, 2008.

[63] *GLPKMEX - A Matlab MEX Interface for the GLPK library, http://glpkmex.sourceforge.net/.*

[64] J.-F. Cordeau, "A branch-and-cut algorithm for the dial-a-ride problem," *Operations Research*, vol. 54, no. 3, pp. 573–586, 2006.

[65] H. Hashimoto, M. Yagiura, S. Imahori, and T. Ibaraki, "Recent progress of local search in handling the time window constraints of the vehicle routing problem," *4OR*, vol. 8, no. 3, pp. 221–238, 2010.

[66] J. K. Lenstra and A. H. G. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, 1981.

[67] C. Expósito-Izquierdo, A. Rossi, and M. Sevaux, "A two-level solution approach to solve the clustered capacitated vehicle routing problem," *Computers & Industrial Engineering*, vol. 91, pp. 274 – 289, 2016.

[68] M. M. S. Haghighi, M. Hadi Zahedi, and M. Ghazizadeh, *A multi level priority clustering GA based approach for solving heterogeneous Vehicle Routing Problem (PCGVRP)*. Dordrecht: Springer Netherlands, 2010, pp. 331–335.

[69] R. Dondo and J. Cerdá, "A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows," *European Journal of Operational Research*, vol. 176, no. 3, pp. 1478 – 1507, 2007.

[70] B. Kallehauge, J. Larsen, O. B. Madsen, and M. M. Solomon, *Vehicle Routing Problem with Time Windows*. Boston, MA: Springer US, 2005, pp. 67–98.

[71] S. Holm and M. M. Sørensen, "The optimal graph partitioning problem," *Operations-Research-Spektrum*, vol. 15, no. 1, pp. 1–8, Mar 1993.

[72] L. Lovász, "Random walks on graphs: A survey," *Combinatorics, Paul erdos is eighty*, vol. 2, no. 1, pp. 1 – 46, 1993.

[73] M. M. Solomon, "VRPTW Benchmark Problems," http://web.cba.neu.edu/~solomon/problems.htm, accessed: 2017-08-30.

[74] U. Graham and I. Cook, *A Dictionary of Statistics*. Oxford University Press, 2008.

[75] J. Larsen, "Parallelization of vehicle routing problem with time windows," Ph.D. dissertation, IMM - DTU Technical University of Denmark, 2001.